

Fondasi dari Model Bahasa Besar

Tong Xiao dan Jingbo Zhu

1626611501.09223v1

17 Januari 2025

Lab NLP, Universitas Northeastern & Penelitian NiuTrans

Hak Cipta © 2021-2025 Tong Xiao dan Jingbo Zhu

**LAB PENGOLAHAN BAHASA ALAMI , NORTHEASTERN UNIVERSITY
&
PENELITIAN NIUTRANS**

Dilisensikan di bawah Lisensi Creative Commons Atribusi-NonKomersial 4.0 Tanpa Porting ("Lisensi"). Anda tidak boleh menggunakan berkas ini kecuali sesuai dengan Lisensi. Anda dapat memperoleh salinan Lisensi di <http://creativecommons.org/licenses/by-nc/4.0>. Kecuali sebagaimana diwajibkan oleh hukum yang berlaku atau disetujui secara tertulis, perangkat lunak yang didistribusikan berdasarkan Lisensi didistribusikan atas dasar "SEBAGAIMANA ADANYA" , TANPA JAMINAN ATAU KETENTUAN APAPUN, tersurat maupun tersirat. Lihat Lisensi untuk bahasa khusus yang mengatur izin dan batasan. di bawah Lisensi.

17 Januari 2025

Kata pengantar

Model bahasa besar berasal dari pemrosesan bahasa alami, tetapi tidak diragukan lagi model ini telah menjadi salah satu kemajuan teknologi paling revolusioner di bidang kecerdasan buatan dalam beberapa tahun terakhir. Wawasan penting yang dibawa oleh model bahasa besar adalah bahwa pengetahuan tentang dunia dan bahasa dapat diperoleh melalui tugas pemodelan bahasa skala besar, dan dengan cara ini, kita dapat membuat model universal yang menangani berbagai masalah. Penemuan ini telah berdampak besar pada metodologi penelitian dalam pemrosesan bahasa alami dan banyak disiplin ilmu terkait. Kita telah beralih dari pelatihan sistem khusus dari awal menggunakan sejumlah besar data berlabel ke paradigma baru menggunakan pra-pelatihan skala besar untuk mendapatkan model dasar, yang kemudian disempurnakan, diselaraskan, dan didorong.

Buku ini bertujuan untuk menguraikan konsep dasar model bahasa besar dan memperkenalkan teknik terkait. Seperti yang tersirat dalam judulnya, buku ini lebih berfokus pada aspek dasar model bahasa besar daripada menyediakan cakupan komprehensif dari semua metode mutakhir.

Buku ini terdiri dari empat bab:

- Bab 1 memperkenalkan dasar-dasar pra-pelatihan. Ini adalah fondasi model bahasa yang besar, dan metode pra-pelatihan umum serta arsitektur model akan dibahas di sini.
- Bab 2 memperkenalkan model generatif, yang merupakan model bahasa besar yang umum kita sebut saat ini. Setelah menyajikan proses dasar pembuatan model ini, kita juga akan mengeksplorasi cara meningkatkan pelatihan model dan menangani teks panjang.
- Bab 3 memperkenalkan metode prompting untuk model bahasa yang besar. Kita akan membahas berbagai strategi prompting, bersama dengan metode yang lebih maju seperti penalaran berantai dan desain prompt otomatis.
- Bab 4 memperkenalkan metode penyelarasan untuk model bahasa yang besar. Bab ini berfokus pada penyempurnaan dan penyelarasan instruksi berdasarkan umpan balik manusia.

Jika pembaca memiliki sedikit latar belakang dalam pembelajaran mesin dan pemrosesan bahasa alami, beserta pemahaman tertentu tentang jaringan saraf seperti Transformers, membaca buku ini akan cukup mudah. Namun, bahkan tanpa pengetahuan sebelumnya ini, buku ini tetap baik-baik saja, karena kami telah membuat konten setiap bab selengkap mungkin, memastikan bahwa pembaca tidak akan terbebani dengan kesulitan membaca yang terlalu banyak.

Dalam penulisan buku ini, kami perlahan menyadari bahwa buku ini lebih seperti kompilasi "catatan" yang kami buat saat mempelajari model bahasa besar. Melalui gaya penulisan catatan ini, kami berharap dapat menawarkan kepada para pembaca jalur pembelajaran yang fleksibel. Apakah mereka ingin mendalami area tertentu atau memperoleh pemahaman yang komprehensif tentang model bahasa besar, mereka akan menemukan pengetahuan dan wawasan yang mereka butuhkan dalam "catatan" ini.

Kami ingin mengucapkan terima kasih kepada para mahasiswa di laboratorium kami dan semua teman yang telah berbagi pandangan mereka tentang model bahasa besar dan membantu mengoreksi kesalahan dalam penulisan. Secara khusus, kami ingin mengucapkan terima kasih kepada Weiqiao Shan, Yongyu Mu, Chenglong Wang, Kaiyan Chang, Yuchun Fan, Hang Zhou, Xinyu Liu, Huiwen Bao, Tong Zheng, Junhao Ruan, dan Qing Yang.

Notasi

sebuah variabel

vektor baris atau matriks

$f(a)$ fungsi dari a

maks $f(a)$ nilai maksimum $f(a)$

arg maxa $f(a)$ nilai a yang memaksimalkan $f(a)$

x urutan token masukan ke model

token masukan x_j pada posisi j

y keluaran urutan token yang diproduksi oleh model

token keluaran y_i pada posisi i

Parameter model \hat{y}

$Pr(a)$ probabilitas suatu

$Pr(a|b)$ probabilitas bersyarat dari b yang diberikan

Distribusi probabilitas $Pr(\cdot|b)$ dari suatu variabel jika diketahui b

$Pr(\hat{y}|a)$ probabilitas a yang diparameterisasi oleh \hat{y}

h_t keadaan tersembunyi pada langkah waktu t dalam model sekuensial

Matriks H dari semua status tersembunyi dari waktu ke waktu dalam suatu urutan

Matriks Q, K, V, kunci, dan nilai dalam mekanisme perhatian

Softmax(A) Fungsi Softmax yang menormalkan vektor input atau matriks A

Fungsi kerugian L

Dataset D digunakan untuk pelatihan atau penyempurnaan model

$\frac{\hat{y}_L}{\hat{y}^*}$ gradien fungsi kerugian L terhadap parameter \hat{y}

$KL(p || q)$ Divergensi KL antara distribusi p dan q

Isi

1 Pra-pelatihan	1
1.1 Pra-pelatihan Model NLP	1
1.1.1 Pra-pelatihan tanpa pengawasan, dengan pengawasan, dan dengan pengawasan mandiri	2
1.1.2 Mengadaptasi Model yang Telah Dilatih	3
1.2 Tugas Pra-Pelatihan yang Diawasi Sendiri	7
1.2.1 Pra-pelatihan hanya dekoder	7
1.2.2 Pra-pelatihan khusus encoder	8
1.2.3 Pra-pelatihan Encoder-Decoder :	15
1.2.4 Perbandingan Tugas Pra-Pelatihan	20
1.3 Contoh: BERT	21
1.3.1 Model Standar	21
1.3.2 Lebih Banyak Pelatihan dan Model yang Lebih Besar	27
1.3.3 Model yang Lebih Efisien	27
1.3.4 Model Multibahasa	28
1.4 Menerapkan Model BERT	30
1.5 Ringkasan	35
2 Model Generatif	36
2.1 Pengantar Singkat tentang LLM	37
2.1.1 Transformator Hanya Dekoder	38
2.1.2 Pelatihan LLM	40
2.1.3 Penyempurnaan LLM	42
2.1.4 Menyelaraskan LLM dengan Dunia :	46
2.1.5 Meminta LLM	51
2.2 Pelatihan dalam Skala Besar	56
2.2.1 Persiapan Data	56
2.2.2 Modifikasi Model	58
2.2.3 Pelatihan Terdistribusi	60
2.2.4 Hukum Skala	63
2.3 Pemodelan Urutan Panjang	66
2.3.1 Optimasi dari Perspektif HPC	67
2.3.2 Arsitektur yang Efisien	68
2.3.3 Cache dan Memori	71
2.3.4 Berbagi antar Kepala dan Lapisan	80

2.3.5 Ekstrapolasi dan Interpolasi Posisi	82
2.3.6 Keterangan	92
2.4 Ringkasan	94
3 Dorongan	96
3.1 Desain Prompt Umum	97
3.1.1 Dasar-dasar	97
3.1.2 Pembelajaran dalam konteks	99
3.1.3 Strategi Rekayasa Cepat	101
3.1.4 Contoh Lainnya	106
3.2 Metode Prompting Lanjutan	115
3.2.1 Rantai Pikiran	115
3.2.2 Dekomposisi Masalah	117
3.2.3 Penyempurnaan diri	124
3.2.4 Ensembling	130
3.2.5 Penggunaan RAG dan Alat	134
3.3 Belajar Memberi Petunjuk	138
3.3.1 Optimasi Prompt	139
3.3.2 Perintah Lunak	142
3.3.3 Pengurangan Panjang Prompt	152
3.4 Ringkasan	153
4 Penyelarasan	155
4.1 Tinjauan Umum Penyelarasan LLM	155
4.2 Penyelarasan Instruksi	157
4.2.1 Penyetelan Halus yang Diawasi	157
4.2.2 Penyempurnaan Akuisisi Data	161
4.2.3 Penyetelan Halus dengan Data Lebih Sedikit	166
4.2.4 Generalisasi Instruksi	167
4.2.5 Menggunakan Model Lemah untuk Memperbaiki Model Kuat	169
4.3 Penyelarasan Preferensi Manusia: RLHF	172
4.3.1 Dasar-dasar Pembelajaran Penguatan	173
4.3.2 Model Penghargaan Pelatihan	179
4.3.3 Pelatihan LLM	182
4.4 Peningkatan Penyelarasan Preferensi Manusia	187
4.4.1 Pemodelan Penghargaan yang Lebih Baik	187

kita	Notasi
4.4.2 Optimasi Preferensi Langsung	193
4.4.3 Pembuatan Data Preferensi Otomatis	196
4.4.4 Penyelarasan Langkah demi Langkah	198
4.4.5 Penyelarasan Waktu Inferensi	200
4.5 Ringkasan	201
Bibliografi	203

BAB 1

Pra-pelatihan

Pengembangan model sekuens neural, seperti Transformers [Vaswani et al., 2017], bersama dengan peningkatan dalam pembelajaran mandiri berskala besar, telah membuka pintu menuju pemahaman dan pembuatan bahasa universal. Pencapaian ini sebagian besar dimotivasi oleh pra-pelatihan: kami memisahkan komponen umum dari banyak sistem berbasis jaringan neural, lalu melatihnya pada sejumlah besar data tak berlabel menggunakan pengawasan mandiri. Model pra-terlatih ini berfungsi sebagai model dasar yang dapat dengan mudah diadaptasi ke berbagai tugas melalui penyempurnaan atau pemberian perintah. Akibatnya, paradigma NLP telah berubah drastis. Dalam banyak kasus, pembelajaran terbimbing berskala besar untuk tugas-tugas tertentu tidak lagi diperlukan, dan sebagai gantinya, kami hanya perlu mengadaptasi model dasar pra-terlatih.

Meskipun pra-pelatihan telah mendapatkan popularitas dalam penelitian NLP baru-baru ini, konsep ini sudah ada sejak beberapa dekade lalu hingga hari-hari awal pembelajaran mendalam. Misalnya, upaya awal untuk melakukan pra-pelatihan sistem pembelajaran mendalam mencakup pembelajaran tanpa pengawasan untuk RNN, jaringan umpan maju dalam, autoencoder, dan lainnya [Schmidhuber, 2015]. Di era pembelajaran mendalam modern, kami mengalami kebangkitan pra-pelatihan, yang sebagian disebabkan oleh pembelajaran tanpa pengawasan skala besar dari berbagai model penyematan kata [Mikolov et al., 2013b; Pennington et al., 2014]. Selama periode yang sama, pra-pelatihan juga menarik minat yang signifikan dalam visi komputer, di mana model tulang punggung dilatih pada kumpulan data berlabel yang relatif besar seperti ImageNet, dan kemudian diterapkan ke berbagai tugas hilir [He et al., 2019; Zoph et al., 2020]. Penelitian skala besar tentang pra-pelatihan dalam NLP dimulai dengan pengembangan model bahasa menggunakan pembelajaran yang diawasi sendiri. Keluarga model ini mencakup beberapa contoh terkenal seperti BERT [Devlin et al., 2019] dan GPT [Brown et al., 2020], semuanya dengan ide serupa bahwa pemahaman dan pembuatan bahasa umum dapat dicapai dengan melatih model untuk memprediksi kata-kata yang ditutupi dalam sejumlah besar teks. Meskipun pendekatan ini sederhana, model yang dihasilkan menunjukkan kemampuan luar biasa dalam memodelkan struktur linguistik, meskipun tidak dilatih secara eksplisit untuk mencapainya. Generalitas tugas pra-pelatihan mengarah pada sistem yang menunjukkan kinerja yang kuat dalam berbagai macam masalah NLP, bahkan mengungguli sistem terbimbing yang dikembangkan dengan baik sebelumnya. Baru-baru ini, model bahasa besar yang telah dilatih sebelumnya telah mencapai keberhasilan yang lebih besar, menunjukkan prospek yang menarik untuk kecerdasan buatan yang lebih umum [Bubeck et al., 2023].

Bab ini membahas konsep pra-pelatihan dalam konteks NLP. Dimulai dengan pengenalan umum tentang metode pra-pelatihan dan aplikasinya. BERT kemudian digunakan sebagai contoh untuk menggambarkan bagaimana model sekuens dilatih melalui tugas yang diawasi sendiri, yang disebut pemodelan bahasa bertopeng. Ini diikuti dengan pembahasan tentang metode untuk mengadaptasi model sekuens yang telah dilatih sebelumnya untuk berbagai tugas NLP. Perhatikan bahwa dalam bab ini, kami akan berfokus terutama pada paradigma pra-pelatihan dalam NLP, dan oleh karena itu, kami tidak bermaksud untuk membahas detail tentang model bahasa besar yang generatif. Pembahasan terperinci tentang model-model ini akan dibahas pada bab-bab berikutnya.

1.1 Pra-pelatihan Model NLP

Pembahasan masalah pra-pelatihan dalam NLP biasanya melibatkan dua jenis masalah: pemodelan urutan (atau pengkodean urutan) dan pembuatan urutan. Meskipun masalah-masalah ini memiliki tujuan yang berbeda,

bentuk-bentuk tersebut, demi kesederhanaan, kami menggambarkannya menggunakan satu model tunggal yang didefinisikan sebagai berikut:

$$\begin{aligned} o &= g(x_0, x_1, \dots, x_m; \hat{y}) = g\hat{y}(x_0, \\ &x_1, \dots, x_m) \end{aligned} \quad (1.1)$$

di mana $\{x_0, x_1, \dots, x_m\}$ menyatakan urutan token input¹, x_0 menyatakan simbol khusus (s atau [CLS]) yang dilampirkan ke awal urutan, $g(\cdot; \hat{y})$ (juga ditulis sebagai $g\hat{y}(\cdot)$) menyatakan jaringan saraf dengan parameter \hat{y} , dan o menyatakan output dari jaringan saraf. Masalah yang berbeda dapat bervariasi berdasarkan bentuk output o . Misalnya, dalam masalah prediksi token (seperti dalam pemodelan bahasa), o adalah distribusi atas kosakata; dalam masalah pengkodean urutan, o adalah representasi dari urutan input, sering kali dinyatakan sebagai urutan vektor bernilai riil.

Ada dua masalah mendasar di sini.

- Mengoptimalkan \hat{y} pada tugas pra-pelatihan. Tidak seperti masalah pembelajaran standar dalam NLP, pra-pelatihan tidak mengasumsikan tugas hilir tertentu yang akan diterapkan pada model tersebut. Sebaliknya, tujuannya adalah untuk melatih model yang dapat digeneralisasikan ke berbagai tugas.
- Menerapkan model pra-latih $g\hat{y}(\cdot)$ ke tugas-tugas hilir. Untuk mengadaptasi model ke tugas-tugas ini, kita perlu sedikit menyesuaikan parameter \hat{y} menggunakan data berlabel atau meminta model dengan deskripsi tugas.

Pada bagian ini, kami membahas ide-ide dasar dalam menangani isu-isu ini.

1.1.1 Pra-pelatihan Tanpa Pengawasan, Dengan Pengawasan dan Dengan Pengawasan Mandiri

Dalam pembelajaran mendalam, pra-pelatihan mengacu pada proses mengoptimalkan jaringan saraf sebelum dilatih/disesuaikan lebih lanjut dan diterapkan pada tugas yang diinginkan. Pendekatan ini didasarkan pada asumsi bahwa model yang telah dilatih sebelumnya pada satu tugas dapat diadaptasi untuk melakukan tugas lain. Akibatnya, kita tidak perlu melatih jaringan saraf yang dalam dan kompleks dari awal pada tugas dengan data berlabel terbatas. Sebaliknya, kita dapat memanfaatkan tugas-tugas yang sinyal supervisinya lebih mudah diperoleh. Hal ini mengurangi ketergantungan pada data berlabel khusus tugas, sehingga memungkinkan pengembangan model yang lebih umum yang tidak terbatas pada masalah tertentu.

Selama kebangkitan jaringan saraf melalui pembelajaran mendalam, banyak upaya awal untuk mencapai pra-pelatihan difokuskan pada pembelajaran tanpa pengawasan. Dalam metode ini, parameter jaringan saraf dioptimalkan menggunakan kriteria yang tidak terkait langsung dengan tugas tertentu. Misalnya, kita dapat meminimalkan entropi silang rekonstruksi vektor masukan untuk setiap lapisan [Bengio et al., 2006]. Pra-pelatihan tanpa pengawasan umumnya digunakan sebagai langkah awal sebelum pembelajaran terbimbing, yang menawarkan beberapa keuntungan, seperti membantu penemuan minimum lokal yang lebih baik dan menambahkan efek regularisasi ke proses pelatihan [Erhan et al., 2010]. Manfaat ini membuat fase pembelajaran terbimbing berikutnya lebih mudah dan lebih stabil.

Pendekatan kedua untuk pra-pelatihan adalah melakukan pra-pelatihan jaringan saraf pada tugas pembelajaran terbimbing. Misalnya, pertimbangkan model urutan yang dirancang untuk mengkodekan urutan masukan ke dalam beberapa

¹Di sini kita berasumsi bahwa token adalah unit dasar teks yang dipisahkan melalui tokenisasi. Terkadang, kita akan menggunakan istilah token dan kata dapat dipertukarkan, meski keduanya memiliki erat tetapi sedikit berbeda maknanya dalam NLP.

representasi. Dalam pra-pelatihan, model ini digabungkan dengan lapisan klasifikasi untuk membentuk sistem klasifikasi. Sistem ini kemudian dilatih pada tugas pra-pelatihan, seperti mengklasifikasikan kalimat berdasarkan sentimen (misalnya, menentukan apakah kalimat menyampaikan sentimen positif atau negatif). Kemudian, kami mengadaptasi model sekuens ke tugas hilir. Kami membangun sistem klasifikasi baru berdasarkan model sekuens yang telah dilatih sebelumnya ini dan lapisan klasifikasi baru (misalnya, menentukan apakah sekuens subjektif atau objektif). Biasanya, kami perlu menyempurnakan parameter model baru menggunakan data berlabel khusus tugas, memastikan model disesuaikan secara optimal untuk bekerja dengan baik pada jenis data baru ini. Model yang telah disempurnakan kemudian digunakan untuk mengklasifikasikan sekuens baru untuk tugas ini. Keuntungan dari prapelatihan terbimbing adalah bahwa proses pelatihan, baik dalam fase prapelatihan atau penyempurnaan, mudah, karena mengikuti paradigma umum pembelajaran terbimbing yang telah dipelajari dengan baik dalam pembelajaran mesin. Namun, seiring meningkatnya kompleksitas jaringan saraf, permintaan akan lebih banyak data berlabel juga meningkat. Hal ini, pada gilirannya, membuat tugas prapelatihan lebih sulit, terutama ketika data berlabel skala besar tidak

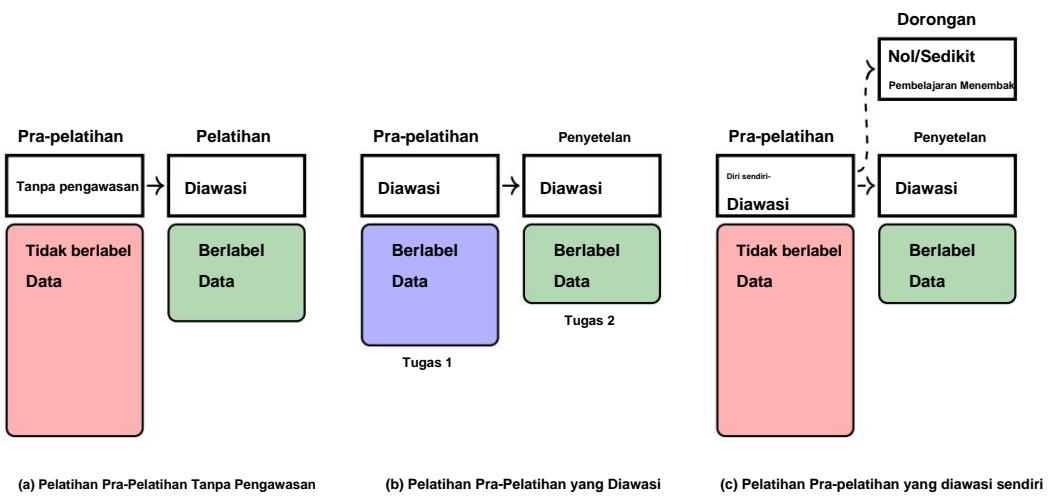
Pendekatan ketiga untuk pra-pelatihan adalah pembelajaran yang diawasi sendiri. Dalam pendekatan ini, jaringan saraf dilatih menggunakan sinyal pengawasan yang dihasilkan oleh dirinya sendiri, bukan yang disediakan oleh manusia. Ini umumnya dilakukan dengan membangun tugas pelatihannya sendiri secara langsung dari data yang tidak berlabel, seperti membuat sistem membuat label semu. Meskipun pembelajaran yang diawasi sendiri baru-baru ini muncul sebagai metode yang sangat populer dalam NLP, ini bukanlah konsep baru. Dalam pembelajaran mesin, konsep terkait adalah pelatihan mandiri di mana model ditingkatkan secara berulang dengan belajar dari label semu yang ditetapkan ke kumpulan data. Untuk melakukan ini, kita memerlukan beberapa data awal untuk membangun model awal. Model ini kemudian menghasilkan label semu untuk data yang tidak berlabel, dan label semu ini selanjutnya digunakan untuk menyempurnakan dan mem-bootstrap model itu sendiri secara berulang. Metode seperti itu telah berhasil digunakan di beberapa area NLP, seperti disambiguasi makna kata [Yarowsky, 1995] dan klasifikasi dokumen [Blum dan Mitchell, 1998]. Tidak seperti metode pelatihan mandiri standar, pra-pelatihan mandiri dalam NLP tidak bergantung pada model awal untuk membuat anotasi data. Sebaliknya, semua sinyal supervisi dibuat dari teks, dan seluruh model dilatih dari awal. Contoh yang terkenal dari hal ini adalah melatih model urutan dengan memprediksi kata yang ditutupi secara berurutan berdasarkan kata-kata sebelumnya atau di sekitarnya dalam sebuah teks. Hal ini memungkinkan pembelajaran mandiri berskala besar untuk jaringan saraf dalam, yang mengarah pada keberhasilan pra-pelatihan dalam banyak tugas pemahaman, penulisan, dan penalaran.

Gambar 1.1 menunjukkan perbandingan dari tiga pendekatan pra-pelatihan di atas. Pra-pelatihan yang diawasi sendiri sangat berhasil sehingga sebagian besar model NLP terkini didasarkan pada paradigma ini. Oleh karena itu, dalam bab ini dan di seluruh buku ini, kami akan fokus pada pra-pelatihan yang diawasi sendiri. Kami akan menunjukkan bagaimana model sekuens dilatih terlebih dahulu melalui pengawasan sendiri dan bagaimana model yang telah dilatih sebelumnya diterapkan.

1.1.2 Mengadaptasi Model yang Telah Dilatih

Seperti disebutkan di atas, dua jenis model utama digunakan secara luas dalam pra-pelatihan NLP.

- **Model Pengodean Urutan.** Dengan adanya urutan kata atau token, model pengodean urutan merepresentasikan urutan ini sebagai vektor bernilai riil atau urutan vektor, dan memperoleh representasi urutan tersebut. Representasi ini biasanya digunakan sebagai masukan untuk model lain, seperti sistem klasifikasi kalimat.



Gbr. 1.1: Ilustrasi pra-pelatihan tanpa pengawasan, dengan pengawasan, dan dengan pengawasan mandiri. Dalam pra-pelatihan tanpa pengawasan, pra-pelatihan dilakukan pada data tak berlabel berskala besar. Hal ini dapat dilihat sebagai langkah awal untuk memiliki titik awal yang baik bagi proses pengoptimalan berikutnya, meskipun upaya yang cukup besar masih diperlukan untuk melatih model lebih lanjut dengan data berlabel setelah pra-pelatihan. Dalam pra-pelatihan dengan pengawasan, asumsi yang mendasarinya adalah bahwa tugas-tugas pembelajaran (dengan pengawasan) yang berbeda saling terkait. Jadi, pertama-tama kita dapat melatih model pada satu tugas, dan mentransfer model yang dihasilkan ke tugas lain dengan beberapa upaya pelatihan atau penyetelan. Dalam pra-pelatihan dengan pengawasan mandiri, model dilatih terlebih dahulu pada data tak berlabel berskala besar melalui pengawasan mandiri. Model dapat dilatih dengan baik dengan cara ini, dan kita dapat secara efisien mengadaptasinya ke tugas-tugas baru melalui penyempurnaan atau dorongan.

- **Model Pembuatan Urutan.** Dalam NLP, pembuatan urutan umumnya mengacu pada masalah pembuatan urutan token berdasarkan konteks tertentu. Istilah konteks memiliki arti yang berbeda di berbagai aplikasi. Misalnya, istilah ini mengacu pada token sebelumnya dalam pemodelan bahasa, dan mengacu pada urutan bahasa sumber dalam penerjemahan mesin²

Kita memerlukan teknik yang berbeda untuk menerapkan model-model ini pada tugas-tugas hilir setelah pra-pelatihan. Di sini kami tertarik pada dua metode berikut.

1.1.2.1 Penyempurnaan Model yang Telah Dilatih Sebelumnya

Untuk pra-pelatihan pengkodean sekuens, metode umum untuk mengadaptasi model yang telah dilatih sebelumnya adalah fine-tuning. Misalkan $\text{Encode}(\cdot)$ menunjukkan encoder dengan parameter θ , misalnya, $\text{Encode}(\cdot)$ dapat menjadi encoder Transformer standar. Asalkan kita telah melatih model ini sebelumnya dengan cara tertentu dan memperoleh parameter optimal θ^* , kita dapat menggunakan其 untuk memodelkan sekuens apa pun dan menghasilkan representasi yang sesuai, seperti ini

$$H = \text{Kode}^\theta(x) \quad (1.2)$$

di mana x adalah urutan masukan $\{x_0, x_1, \dots, x_m\}$, dan H adalah representasi keluaran yang merupakan urutan vektor bernali riil $\{h_0, h_1, \dots, h_m\}$. Karena encoder tidak bekerja sebagai sistem NLP yang berdiri sendiri, ia sering kali diintegrasikan sebagai komponen ke dalam sistem yang lebih besar. Pertimbangkan, misalnya, masalah klasifikasi teks di mana kita mengidentifikasi polaritas (yaitu, positif, negatif,

²Lebih tepatnya, dalam decoding auto-regresif terjemahan mesin, setiap token bahasa sasaran dibuat berdasarkan token sebelumnya dan urutan bahasa sumber.

dan netral) dari suatu teks. Kita dapat membangun sistem klasifikasi teks dengan menumpuk pengklasifikasi di atas encoder. Misalkan $\text{Classify}^y(\cdot)$ adalah jaringan saraf dengan parameter y . Maka, model klasifikasi teks dapat dinyatakan dalam bentuk

$$\begin{aligned} \Pr y, \hat{y}(-|x) &= \text{Klasifikasi}^y(H) \\ &= \text{Klasifikasi}^y(\text{Encode}^{\hat{y}}(x)) \end{aligned} \quad (1.3)$$

Di sini $\Pr y, \hat{y}(-|x)$ adalah distribusi probabilitas atas himpunan label {positif, negatif, netral}, dan label dengan probabilitas tertinggi dalam distribusi ini dipilih sebagai keluaran. Untuk menjaga notasi tetap rapi, kita akan menggunakan F

$\hat{y}, \tilde{y}(\cdot)$ untuk menyatakan $\text{Klasifikasi}^y(\text{Encode}^{\hat{y}}(\cdot))$.

Karena parameter model y dan \hat{y} tidak dioptimalkan untuk tugas klasifikasi, kita tidak dapat langsung menggunakan model ini. Sebaliknya, kita harus menggunakan versi model yang dimodifikasi yang disesuaikan dengan tugas tersebut. Cara yang umum adalah menyempurnakan model dengan memberikan pelabelan eksplisit dalam tugas-tugas hilir.

Kita dapat melatih $F\hat{y}, \tilde{y}(\cdot)$ pada kumpulan data berlabel, memperlakukannya sebagai tugas pembelajaran terbimbing umum. Hasil dari penyempurnaan adalah parameter \hat{y} dan \tilde{y} yang dioptimalkan lebih lanjut. Atau, kita dapat membekukan parameter encoder \hat{y} untuk mempertahankan status pra-latihan, dan hanya berfokus pada pengoptimalan y . Hal ini memungkinkan pengklasifikasi untuk secara efisien disesuaikan agar dapat bekerja bersama-sama dengan encoder pra-latih.

Setelah kita memperoleh model yang telah disempurnakan, kita dapat menggunakannya untuk mengklasifikasikan teks baru. Misalnya, anggaplah kita memiliki komentar yang diposting di situs web perjalanan:

Saya suka makanan di sini. Luar biasa!

Pertama-tama kita tokenisasi teks ini menjadi x_1, x_2, \dots, x_n dan kemudian masukkan urutan token x_{new} ke dalam fine-tuned token3 model $F\hat{y}, \tilde{y}(\cdot)$. Model ini menghasilkan distribusi ke seluruh kelas dengan

$$F\hat{y}, \tilde{y}(x_{baru}) = \Pr(\text{positif}|x_{baru}) \Pr(\text{negatif}|x_{baru}) \Pr(\text{netral}|x_{baru}) \quad (1.4)$$

Dan kita pilih label entri dengan nilai maksimum sebagai output. Dalam contoh ini, nilainya positif.

Secara umum, jumlah data berlabel yang digunakan dalam fine-tuning lebih kecil dibandingkan dengan data pra-pelatihan, sehingga fine-tuning lebih murah secara komputasi. Hal ini membuat adaptasi model pra-pelatihan sangat efisien dalam praktik: dengan model pra-pelatihan dan tugas hilir, kita hanya perlu mengumpulkan beberapa data berlabel, dan sedikit menyesuaikan parameter model pada data ini. Pembahasan lebih rinci tentang fine-tuning dapat ditemukan di Bagian 1.4.

1.1.2.2 Prompting Model yang Telah Dilatih Sebelumnya

Tidak seperti model pengkodean sekvensi, model pembangkitan sekvensi sering digunakan secara independen untuk mengatasi masalah pembangkitan bahasa, seperti menjawab pertanyaan dan penerjemahan mesin, tanpa memerlukan modul tambahan. Oleh karena itu, mudah untuk menyempurnakan model ini.

3Teks dapat ditokenisasi dengan berbagai cara. Salah satu yang paling mudah adalah dengan membagi teks menjadi kata-kata dan tanda baca dalam bahasa Inggris {I, love, the, food, here, ..It, 's, amazing, !}

sebagai sistem yang lengkap pada tugas-tugas hilir. Misalnya, kita dapat menyempurnakan model multibahasa encoder-decoder yang telah dilatih sebelumnya pada beberapa data bilingual untuk meningkatkan kinerjanya pada tugas penerjemahan tertentu.

Di antara berbagai model pembangkitan sekuens, contoh yang menonjol adalah model bahasa besar yang dilatih pada sejumlah besar data. Model bahasa ini dilatih untuk sekadar memprediksi token berikutnya berdasarkan token sebelumnya. Meskipun prediksi token merupakan tugas yang sangat sederhana sehingga telah lama dibatasi hanya pada "pemodelan bahasa", prediksi token telah terbukti memungkinkan pembelajaran pengetahuan umum bahasa dengan mengulang tugas tersebut berkali-kali. Hasilnya adalah model bahasa besar yang telah dilatih sebelumnya menunjukkan kemampuan yang sangat baik dalam prediksi token, sehingga memungkinkan untuk mengubah banyak masalah NLP menjadi masalah pembangkitan teks sederhana melalui permintaan model bahasa besar. Misalnya, kita dapat membingkai masalah klasifikasi teks di atas sebagai tugas pembangkitan teks

Saya suka makanan di sini. Luar biasa! Saya_____

Di sini menunjukkan kata atau frasa yang ingin kita prediksi (sebut saja pelengkapan). Jika kata yang diprediksi adalah happy, atau glad, atau pleased atau kata positif terkait, kita dapat menggolongkan teks tersebut sebagai positif. Contoh ini menunjukkan metode prompting sederhana di mana kita menggabungkan teks input dengan I'm untuk membentuk prompt. Kemudian, pelengkapan membantu memutuskan label mana yang ditetapkan pada teks asli.

Mengingat kinerja pemahaman bahasa dan pembuatan model bahasa yang besar, perintah dapat memerintahkan model untuk melakukan tugas yang lebih rumit. Berikut adalah perintah di mana kami memerintahkan LLM untuk melakukan klasifikasi polaritas dengan instruksi.

Asumsikan bahwa polaritas teks adalah label yang dipilih dari {positif, negatif, netral}.
Identifikasi polaritas input.

Masukan: Saya suka makanan di sini. Luar biasa!

Polaritas: _____

Dua kalimat pertama merupakan deskripsi tugas. **Input** dan **Polaritas** masing-masing merupakan indikator input dan output. Kami berharap model tersebut dapat melengkapi teks dan sekaligus memberikan label polaritas yang benar. Dengan menggunakan perintah berbasis instruksi, kami dapat mengadaptasi model bahasa yang besar untuk memecahkan masalah NLP tanpa memerlukan pelatihan tambahan.

Contoh ini juga menunjukkan kemampuan pembelajaran zero-shot dari model bahasa besar, yang dapat melakukan tugas yang tidak diamati selama fase pelatihan. Metode lain untuk mengaktifkan kemampuan baru dalam jaringan saraf adalah pembelajaran few-shot. Hal ini biasanya dicapai melalui pembelajaran dalam konteks (TIK). Lebih khusus lagi, kami menambahkan beberapa contoh yang menunjukkan bagaimana masukan berhubungan dengan keluaran. Contoh-contoh ini, yang dikenal sebagai demonstrasi, digunakan untuk mengajarkan model bahasa besar cara melakukan tugas. Berikut adalah contoh yang melibatkan demonstrasi

1.2 Tugas Pra-Pelatihan yang Diawasi Sendiri

7

Asumsikan polaritas teks adalah label yang dipilih dari {positif, negatif, netral}. Identifikasi polaritas input.

Input: Lalu lintas sangat buruk pada jam-jam sibuk, sehingga sulit untuk mencapai bandara tepat waktu.

Polaritas: Negatif

Masukan: Cuaca di sini bagus sekali.

Polaritas: Positif

Masukan: Saya suka makanan di sini. Luar biasa!

Polaritas: _____

Dorongan dan pembelajaran dalam konteks memainkan peran penting dalam peningkatan penggunaan bahasa besar akhir-akhir ini. model. Kami akan membahas masalah ini lebih dalam di Bab 3. Namun, perlu dicatat bahwa meskipun dorongan adalah cara yang ampuh untuk mengadaptasi model bahasa yang besar, beberapa upaya penyesuaian masih diperlukan untuk memastikan model dapat mengikuti instruksi secara akurat. Selain itu, fine-tuning Proses ini sangat penting untuk menyelaraskan nilai-nilai model ini dengan nilai-nilai kemanusiaan. Pembahasan mengenai fine-tuning dapat ditemukan pada Bab 4.

1.2 Tugas Pra-Pelatihan yang Diawasi Sendiri

Pada bagian ini, kami mempertimbangkan pendekatan pra-pelatihan yang diawasi sendiri untuk berbagai arsitektur saraf, termasuk arsitektur khusus dekoder, khusus enkoder, dan enkoder-dekoder. Kami membatasi diri pada pendekatan pra-pelatihan yang diawasi sendiri untuk berbagai arsitektur saraf, termasuk arsitektur khusus dekoder, khusus enkoder, dan enkoder-dekoder. diskusi tentang Transformers karena mereka membentuk dasar dari sebagian besar model pra-pelatihan di NLP. Namun, pra-pelatihan adalah konsep yang luas, jadi kami hanya memberikan pengantar singkat tentang pendekatan dasar untuk membuat bagian ini ringkas.

1.2.1 Pra-pelatihan hanya dekoder

Arsitektur decoder-only telah banyak digunakan dalam pengembangan model bahasa [Radford et al., 2018]. Misalnya, kita dapat menggunakan dekoder Transformer sebagai model bahasa hanya dengan menghapus lapisan-lapisan perhatian silang darinya. Model seperti itu memprediksi distribusi token pada suatu posisi diberikan token sebelumnya, dan outputnya adalah token dengan probabilitas maksimum. Cara standar untuk melatih model ini, seperti dalam masalah pemodelan bahasa, adalah meminimalkan fungsi kerugian. atas kumpulan urutan token. Biarkan Decoder \hat{y} (-) menunjukkan dekoder dengan parameter \hat{y} . Pada setiap posisi i , dekoder menghasilkan distribusi token berikutnya berdasarkan token sebelumnya token $\{x_0, \dots, x_i\}$, dilambangkan dengan $\text{Pr}_{\hat{y}}(\cdot | x_0, \dots, x_i)$ (atau $p_{\hat{y}}(x_{i+1} | x_0, \dots, x_i)$ Singkatnya). Misalkan kita memiliki emas- $i+1$ emas distribusi standar pada posisi yang sama, dilambangkan dengan $p_{\text{emas}}(x_{i+1} | x_0, \dots, x_i)$. Untuk pemodelan bahasa, kita dapat berpikir dari $p_{\text{emas}}^{\text{saya}}$ sebagai representasi one-hot dari kata prediksi yang benar. Kami kemudian mendefinisikan fungsi kerugian $L(p_{\text{saya}}^{\text{emas}}, p_{\text{emas}}^{\text{saya}})$ untuk mengukur perbedaan antara prediksi model dan prediksi sebenarnya. NLP, kerugian entropi silang skala log biasanya digunakan.

Diberikan urutan m token $\{x_0, \dots, x_m\}$, kerugian pada urutan ini adalah jumlah kerugian

atas posisi $\{0, \dots, m \geq 1\}$, diberikan oleh

$$\begin{aligned}
 \text{Kerugian}_y(x_0, \dots, x_m) &= \sum_{i=0}^{m-1} \text{LogCrossEntropy}(x_i, p_{i+1, p_{i+1} \text{ emas}}) \\
 &= \sum_{i=0}^{m-1} \text{LogEntropiSilang}(p_{i+1, p_{i+1} \text{ emas}}) \quad (1.5)
 \end{aligned}$$

di mana $\text{LogCrossEntropy}(\cdot)$ adalah entropi silang skala log, dan p dari saya_{i+1} adalah representasi yang paling panas x_{i+1} .

Fungsi kerugian ini dapat diperluas ke sekumpulan urutan D . Dalam kasus ini, tujuan dari pra-pelatihan adalah untuk menemukan parameter terbaik yang meminimalkan kerugian pada D

$$\hat{y} = \arg \min_{\hat{y}} \sum_{x \in D} \text{Kerugian}_y(x) \quad (1.6)$$

Perlu dicatat bahwa tujuan ini secara matematis setara dengan estimasi kemungkinan maksimum, dan dapat dinyatakan kembali sebagai

$$\begin{aligned}
 \hat{y} &= \arg \max_{\hat{y}} \log \Pr(\hat{y}|x) \\
 &= \arg \max_{\hat{y}} \sum_{x \in D} \log \Pr(\hat{y}|x_i) \quad (1.7)
 \end{aligned}$$

Dengan parameter yang dioptimalkan \hat{y} , kita dapat menggunakan model bahasa terlatih Decoder $^y(\cdot)$ untuk menghitung probabilitas $\Pr(\hat{y}|x_i)$ di setiap posisi deret tertentu.

1.2.2 Pra-pelatihan khusus encoder

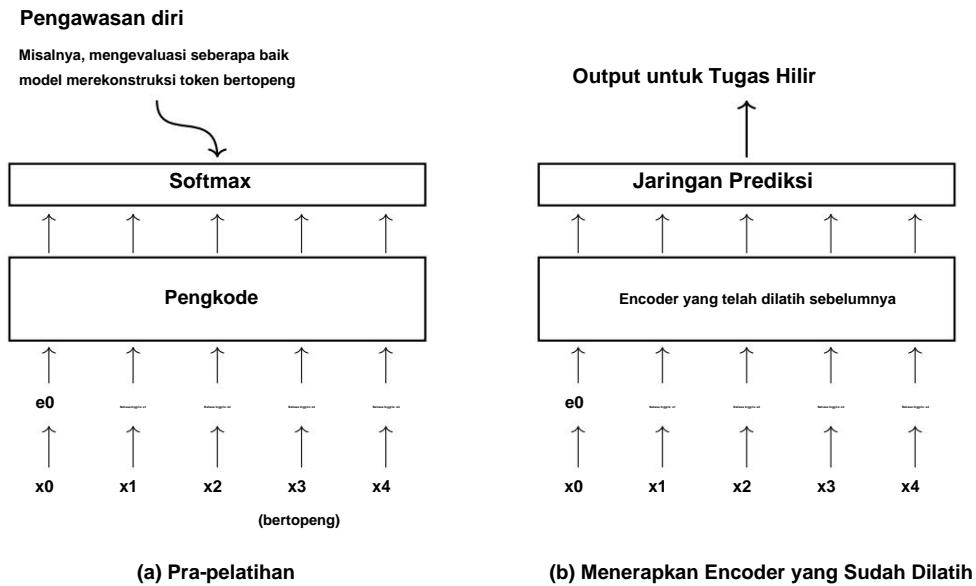
Seperti yang didefinisikan dalam Bagian 1.1.2.1, sebuah encoder $\text{Encoder}_y(\cdot)$ adalah sebuah fungsi yang membaca urutan token $x = x_0 \dots x_m$ dan menghasilkan urutan vektor $H = h_0 \dots h_m$ tidak mudah,⁴ Pelatihan model ini adalah karena kita tidak memiliki data standar emas untuk mengukur seberapa baik keluaran fungsi bernilai riil. Pendekatan umum untuk pra-pelatihan encoder adalah menggabungkan encoder dengan beberapa lapisan keluaran untuk menerima sinyal supervisi yang lebih mudah diperoleh. Gambar 1.2 menunjukkan arsitektur umum untuk pra-pelatihan encoder Transformer, di mana kita menambahkan lapisan Softmax di atas encoder Transformer. Jelas, arsitektur ini sama dengan model bahasa berbasis dekoder, dan keluarannya adalah urutan distribusi probabilitas

$$\begin{matrix}
 & W_y p \\
 -1 & \vdots & \vdots & = \text{Softmax}_W(\text{Pengode}_y(x)) \\
 & pW_y & \vdots &
 \end{matrix} \quad (1.9)$$

⁴ Jika kita melihat h_i sebagai vektor baris, H dapat ditulis sebagai

$$H = \begin{matrix} h_0 \\ \vdots \\ h_m \end{matrix} \quad (1.8)$$

1.2 Tugas Pra-Pelatihan yang Diawasi Sendiri



Gambar 1.2: Pra-pelatihan encoder Transformer (kiri) dan kemudian menerapkan encoder yang telah dilatih sebelumnya (kanan). Dalam pra-pelatihan fase, encoder, bersama dengan lapisan Softmax, dilatih melalui pengawasan mandiri. Dalam fase aplikasi, Softmax lapisan dihapus, dan encoder yang telah dilatih sebelumnya digabungkan dengan jaringan prediksi untuk mengatasi masalah tertentu. Secara umum, agar lebih mudah beradaptasi dengan tugas-tugas tersebut, sistem disempurnakan menggunakan data berlabel.

Di sini $p_i^{W,\hat{y}}$ adalah distribusi keluaran $\Pr(\cdot|x)$ pada posisi i . Kami menggunakan $\text{Softmax}(W \cdot \cdot)$ untuk menunjukkan bahwa Lapisan Softmax diparameterisasi oleh W , yaitu $\text{Softmax}(W(H)) = \text{Softmax}(H \cdot W)$. Untuk notasi sederhana, kita kadang-kadang akan menghilangkan superskrip W dan \hat{y} yang ditempelkan pada setiap probabilitas distribusi.

Perbedaan antara model ini dan model bahasa standar adalah bahwa output p_i memiliki makna yang berbeda dalam pra-pelatihan encoder dan pemodelan bahasa. Dalam pemodelan bahasa, p_i adalah distribusi probabilitas untuk memprediksi kata berikutnya. Ini mengikuti decoding auto-regresif proses: model bahasa hanya mengamati kata-kata hingga posisi i dan memprediksi kata berikutnya. Sebaliknya, dalam pra-pelatihan encoder, seluruh urutan dapat diamati sekaligus, sehingga tidak ada masuk akal untuk memprediksi token apa pun dalam urutan ini.

1.2.2.1 Pemodelan Bahasa Bertopeng

Salah satu metode pra-pelatihan encoder yang paling populer adalah pemodelan bahasa bertopeng, yang membentuk dasar dari model BERT yang terkenal [Devlin et al., 2019]. Ide pemodelan bahasa bertopeng adalah untuk menciptakan tantangan prediksi dengan menutupi beberapa token dalam input urutan dan pelatihan model untuk memprediksi token yang ditutupi. Dalam pengertian ini, masalah pemodelan bahasa konvensional, yang terkadang disebut pemodelan bahasa kausal, adalah kasus khusus. pemodelan bahasa bertopeng: pada setiap posisi, kita menutupi token dalam konteks kanan, dan memprediksi token pada posisi ini menggunakan konteks kirinya. Namun, dalam pemodelan bahasa kausal kita hanya menggunakan konteks kiri dalam prediksi kata, sedangkan prediksi mungkin bergantung pada token dalam konteks yang tepat. Sebaliknya, dalam pemodelan bahasa bertopeng, semua token yang tidak bertopeng digunakan untuk prediksi kata, mengarah ke model dua arah yang membuat prediksi berdasarkan kiri dan konteks yang tepat.

Lebih formalnya, untuk urutan input $x = x_0 \dots x_m$, misalkan kita menutupi token pada posisi $A(x) = \{i_1, \dots, i_u\}$. Maka kita memperoleh urutan token bertopeng x^- di mana token pada setiap posisi di $A(x)$ diganti dengan simbol khusus [MASK]. Misalnya, untuk yang berikut ini urutan

Burung yang bangun pagi akan mendapat cacing

kita mungkin memiliki urutan token bertopeng seperti ini

Burung [MASK] menangkap [MASK]

di mana kita menutupi token awal dan worm (yaitu, $i_1 = 2$ dan $i_2 = 6$).

Sekarang kita memiliki dua urutan x dan x^- . Model tersebut kemudian dioptimalkan sehingga kita dapat memprediksi x dengan benar berdasarkan x^- . Hal ini dapat dianggap sebagai proses seperti autoencoding, dan tujuan pelatihan adalah untuk memaksimalkan probabilitas rekonstruksi $\Pr(x|x^-)$. Perhatikan bahwa ada penyelarasan posisi-bijaksana sederhana antara x dan x^- . Karena token yang tidak bertopeng di x^- sama dengan token-to di x pada posisi yang sama, tidak perlu mempertimbangkan prediksi untuk token yang tidak bertopeng ini. Hal ini mengarah pada tujuan pelatihan yang disederhanakan yang hanya memaksimalkan probabilitas untuk token yang ditutupi. Kita dapat mengekspresikan tujuan ini dalam bentuk estimasi kemungkinan maksimum.

$$(W^-, \hat{y}) = \arg \max (x_i | x^-) \quad \text{logaritma } \Pr_{W, \hat{y}} \quad (1.10)$$

Tentukan nilai $\sum_{i=1}^m \Pr_{W, \hat{y}}(x_i | x^-)$

atau sebagai alternatif mengekspresikannya menggunakan kerugian entropi silang

$$(W, \hat{y}) = \arg \min \left[-\sum_{i=1}^m \Pr_{W, \hat{y}}(x_i | x^-) \log \Pr_{W, \hat{y}}(x_i | x^-) \right] \quad (1.11)$$

di mana $\Pr_{W, \hat{y}}(x_k | x^-)$ adalah probabilitas token x_k yang benar pada posisi k jika input rusak adalah distribusi x , dan $\Pr_{W, \hat{y}}(x_k | x^-)$ probabilitas pada posisi k jika input rusak x . Sebagai ilustrasi, perhatikan contoh di atas di mana dua token dari urutan "burung pagi menangkap cacing" ditutupi. Untuk contoh ini, tujuannya adalah untuk memaksimalkan jumlah probabilitas skala logaritma

$$\text{Kerugian} = \log \Pr(x_2 = \text{awal} | x = [\text{CLS}] \text{ Burung [MASK] menangkap [MASK]}) +$$

$$\log \Pr(x_6 = \text{cacing} | x = [\text{CLS}] \text{ Burung [MASK] menangkap [MASK]}) \quad (1.12)$$

Setelah kita memperoleh parameter W dan \hat{y} yang sudah dioptimalkan, kita dapat menghilangkan W . Kemudian, kita dapat lebih jauh menyempurnakan encoder yang telah dilatih sebelumnya Encoder \hat{y} (-) atau langsung menerapkannya ke tugas hilir.

1.2.2.2 Pemodelan Bahasa Permutasi

Meskipun pemodelan bahasa bertopeng sederhana dan diterapkan secara luas, hal ini menimbulkan masalah baru. Salah satu kelemahannya adalah penggunaan token khusus, [MASK], yang hanya digunakan selama pelatihan tetapi tidak

pada saat pengujian. Hal ini menyebabkan perbedaan antara pelatihan dan inferensi. Selain itu, proses pengkodean otomatis mengabaikan ketergantungan antara token yang ditutupi. Misalnya, pada contoh di atas misalnya, prediksi x_2 (yaitu, token bertopeng pertama) dibuat secara independen dari x_6 (yaitu, token bertopeng kedua), meskipun x_6 harus dipertimbangkan dalam konteks x_2 .

Masalah-masalah ini dapat diatasi dengan menggunakan pendekatan pemodelan bahasa permutasi untuk pra-pelatihan [Yang et al., 2019]. Mirip dengan pemodelan bahasa kausal, pemodelan bahasa permutasi melibatkan pembuatan prediksi token secara berurutan. Namun, tidak seperti pemodelan kausal di mana prediksi mengikuti urutan alami teks (seperti kiri ke kanan atau kanan ke kiri), bahasa yang diubah-ubah memungkinkan prediksi dalam urutan apa pun. Pendekatannya sederhana: kami menentukan memesan prediksi token dan kemudian melatih model dalam cara pemodelan bahasa standar, seperti dijelaskan di Bagian 1.2.1. Perhatikan bahwa dalam pendekatan ini, urutan token sebenarnya dalam teks tetap tidak berubah, dan hanya urutan prediksi token ini yang berbeda dari bahasa standar pemodelan. Misalnya, perhatikan urutan 5 token $x_0x_1x_2x_3x_4$. Biarkan e_i mewakili penempatan token x_i (yaitu, kombinasi penempatan token dan penempatan posisional). Dalam standar pemodelan bahasa, kita akan menghasilkan urutan ini dalam urutan $x_0 \ddot{y} x_1 \ddot{y} x_2 \ddot{y} x_3 \ddot{y} x_4$. Probabilitas urutan dapat dimodelkan melalui proses pembangkitan.

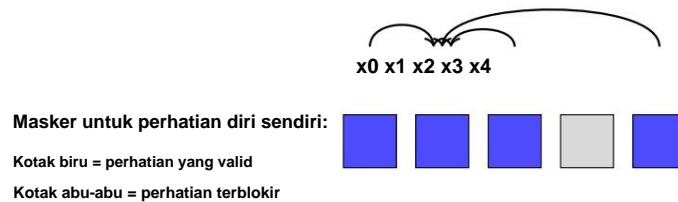
$$\begin{aligned} \Pr(x) &= \Pr(x_0) \cdot \Pr(x_1|x_0) \cdot \Pr(x_2|x_0, x_1) \cdot \Pr(x_3|x_0, x_1, x_2) \cdot \\ &\quad \Pr(x_4|x_0, x_1, x_2, x_3) \\ &= \Pr(x_0) \cdot \Pr(x_1|e_0) \cdot \Pr(x_2|e_0, e_1) \cdot \Pr(x_3|e_0, e_1, e_2) \cdot \\ &\quad \Pr(x_4|e_0, e_1, e_2, e_3) \end{aligned} \tag{1.13}$$

Sekarang, mari kita pertimbangkan urutan prediksi token yang berbeda: $x_0 \ddot{y} x_4 \ddot{y} x_2 \ddot{y} x_1 \ddot{y} x_3$. Proses pembangkitan sekvensi kemudian dapat dinyatakan sebagai berikut:

$$\begin{aligned} \Pr(x) &= \Pr(x_0) \cdot \Pr(x_4|e_0) \cdot \Pr(x_2|e_0, e_4) \cdot \Pr(x_1|e_0, e_4, e_2) \cdot \\ &\quad \Pr(x_3|e_0, e_4, e_2, e_1) \end{aligned} \tag{1.14}$$

Urutan prediksi baru ini memungkinkan pembuatan beberapa token dikondisikan pada konteks yang lebih luas, daripada hanya terbatas pada token sebelumnya seperti dalam bahasa standar model. Misalnya, dalam menghasilkan x_3 , model mempertimbangkan konteks kirinya (yaitu, e_0, e_1, e_2) dan konteks kanan (yaitu, e_4). Penanaman e_0, e_1, e_2, e_4 menggabungkan informasi posisi dari x_0, x_1, x_2, x_4 , dengan tetap mempertahankan urutan token aslinya. Oleh karena itu, pendekatan ini agak mirip dengan pemodelan bahasa bertopeng: kita menutupi x_3 dan menggunakan token di sekitarnya x_0, x_1, x_2, x_4 untuk memprediksi token ini.

Implementasi model bahasa yang dipermutasi relatif mudah bagi Transformers. Karena model perhatian-diri tidak peka terhadap urutan masukan, kita tidak perlu secara eksplisit susun ulang urutan tersebut agar memiliki faktorisasi seperti persamaan (1.14). Sebagai gantinya, permutasi dapat dilakukan dengan menetapkan topeng yang sesuai untuk perhatian diri sendiri. Misalnya, pertimbangkan kasus komputasi $\Pr(x_1|e_0, e_4, e_2)$. Kita dapat mengurutkan x_0, x_1, x_2, x_3, x_4 dan mengalihkan perhatian dari x_3 ke x_1 dalam perhatian diri sendiri, seperti yang dilustrasikan di bawah ini



Untuk contoh yang lebih ilustratif, kami membandingkan hasil penyembunyian perhatian diri dari bahasa kausal pemodelan, pemodelan bahasa bertopeng, dan pemodelan bahasa permutasi pada Gambar 1.3.

1.2.2.3 Pra-pelatihan Encoder sebagai Pengklasifikasi

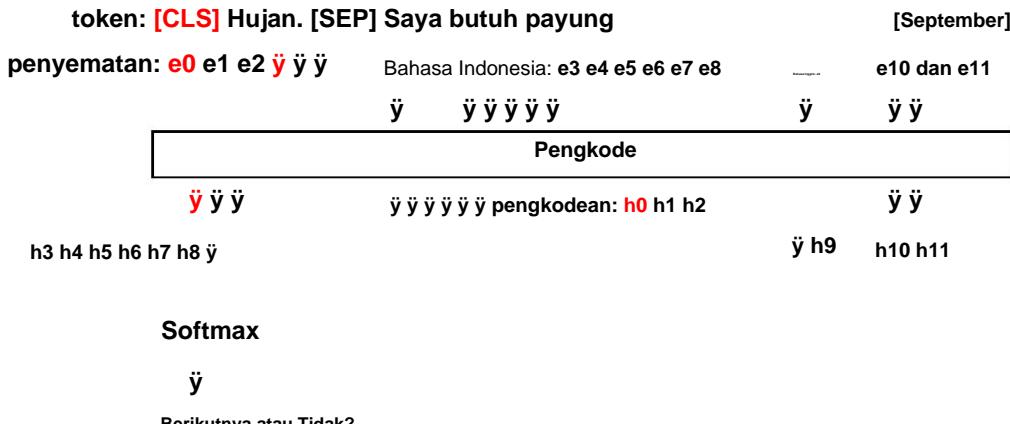
Ide lain yang umum digunakan untuk melatih encoder adalah mempertimbangkan tugas klasifikasi.

Dalam pembelajaran mandiri, hal ini biasanya dilakukan dengan membuat tantangan klasifikasi baru dari teks yang tidak diberi label. Ada banyak cara berbeda untuk merancang tugas klasifikasi. Berikut ini kami menyajikan tugas-tugas populer.

Sebuah metode sederhana, yang disebut prediksi kalimat berikutnya (NSP), disajikan dalam BERT asli kertas [Devlin et al., 2019]. Asumsi NSP adalah bahwa encoder teks yang baik harus menangkap hubungan antara dua kalimat. Untuk memodelkan hubungan seperti itu, dalam NSP kita dapat menggunakan keluaran dari pengkodean dua kalimat berurutan SentA dan SentB untuk menentukan apakah SentB kalimat berikutnya setelah SentA. Misalkan SentA = 'Hujan.' dan SentB = 'Saya butuh payung.' Urutan masukan encoder bisa berupa

[CLS] Sedang hujan. [SEP] Saya butuh payung. [SEP]

di mana [CLS] adalah simbol awal (yaitu, x_0) yang umum digunakan dalam pra-pelatihan encoder, dan [SEP] adalah pemisah yang memisahkan dua kalimat. Pemrosesan urutan ini mengikuti prosedur standar pengkodean Transformer: pertama-tama kita mewakili setiap token x_i sebagai token yang sesuai menamakan e_i , dan kemudian memasukkan urutan penanaman $\{e_0, \dots, e_m\}$ ke dalam encoder untuk mendapatkan urutan keluaran $\{h_0, \dots, h_m\}$. Karena h_0 umumnya dianggap sebagai representasi keseluruhan urutan, kami menambahkan lapisan Softmax di atasnya untuk membangun sistem klasifikasi biner. Ini Prosesnya diilustrasikan sebagai berikut



	x_0	x_1	x_2	x_3	x_4		
x_0	Blue	Grey	Grey	Grey	Grey	→	Persamaan ($x_0 = 1$)
x_1	Blue	Blue	Grey	Grey	Grey	→	Contoh soal $\Pr(x_1 e_0)$
x_2	Blue	Blue	Blue	Grey	Grey	→	Tentukan nilai $x_2 e_0, e_1$
x_3	Blue	Blue	Blue	Blue	Grey	→	$\Pr(x_3 e_0, e_1, e_2)$
x_4	Blue	Blue	Blue	Blue	Blue	→	$\Pr(x_4 e_0, e_1, e_2, e_3)$

(a) Pemodelan Bahasa Kausal (urutan: $x_0 \ddot{\vee} x_1 \ddot{\vee} x_2 \ddot{\vee} x_3 \ddot{\vee} x_4$)

	bertopengbertopeng						
	x_0	x_1	x_2	x_3	x_4		
x_0	Blue	Blue	Blue	Blue	Blue	→	1
x_1 bertopeng	Blue	Blue	Blue	Blue	Blue	→	$\Pr(x_1 e_0, emask, e2, emask, e4)$
x_2	Blue	Blue	Blue	Blue	Blue	→	1
x_3 bertopeng	Blue	Blue	Blue	Blue	Blue	→	$\Pr(x_3 e_0, emas, e2, emas, e4)$
x_4	Blue	Blue	Blue	Blue	Blue	→	1

(b) Pemodelan Bahasa Bertopeng (urutan: $x_0, [MASK], x_2, [MASK], x_4 \ddot{\vee} x_1, x_3$)

	x_0	x_1	x_2	x_3	x_4		
x_0	Blue	Grey	Grey	Grey	Grey	→	Persamaan ($x_0 = 1$)
x_1	Blue	Blue	Blue	Grey	Blue	→	$\Pr(x_1 e_0, e_4, e_2)$
x_2	Blue	Grey	Blue	Grey	Blue	→	Tentukan nilai $x_2 e_0, e_4$
x_3	Blue	Blue	Blue	Blue	Blue	→	$\Pr(x_3 e_0, e_4, e_2, e_1)$
x_4	Blue	Grey	Grey	Grey	Blue	→	Contoh soal $\Pr(x_4 e_0)$

(c) Pemodelan Bahasa Permutasi (urutan: $x_0 \ddot{\vee} x_4 \ddot{\vee} x_2 \ddot{\vee} x_1 \ddot{\vee} x_3$)

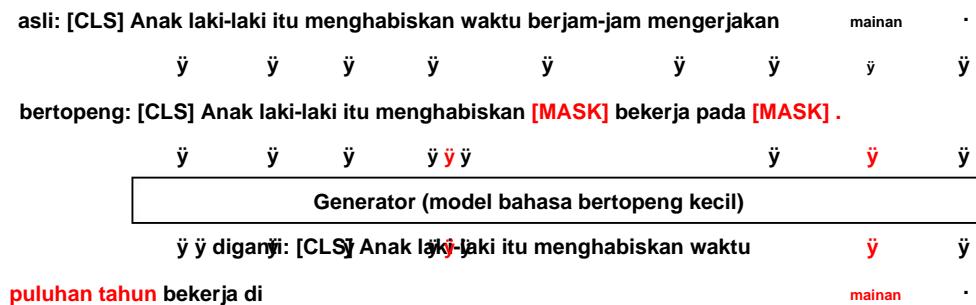
Gambar 1.3: Perbandingan hasil masking perhatian-diri dari pemodelan bahasa kausal, pemodelan bahasa bertopeng, dan pemodelan bahasa permutasi. Sel abu-abu menunjukkan token pada posisi j tidak memperhatikan token pada posisi i .

Sel biru (i, j) menunjukkan bahwa token pada posisi j memperhatikan token pada posisi i . emask mewakili penanaman simbol [MASK], yang merupakan kombinasi dari penanaman token dan penanaman posisi.

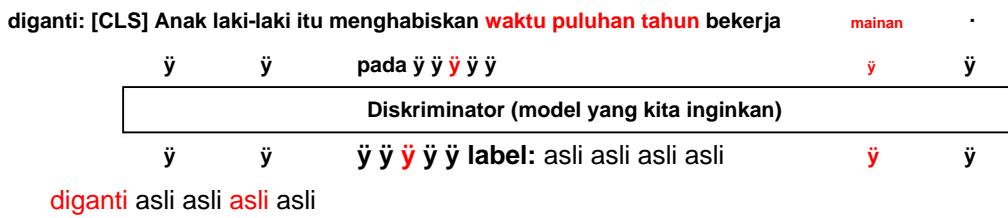
Untuk menghasilkan sampel pelatihan, kita memerlukan dua kalimat setiap kali, satu untuk SentA dan yang lainnya untuk SentB. Cara sederhana untuk melakukannya adalah dengan memanfaatkan urutan alami dari dua kalimat berurutan dalam teks. Misalnya, kita memperoleh sampel positif dengan menggunakan kalimat berurutan yang sebenarnya, dan sampel negatif dengan menggunakan kalimat yang diambil sampelnya secara acak. Akibatnya, melatih model ini sama dengan melatih pengklasifikasi. Biasanya, NSP digunakan sebagai ker-

berfungsi untuk pra-pelatihan berdasarkan pemodelan bahasa tertopeng.

Contoh kedua dari pelatihan encoder Transformer sebagai pengklasifikasi adalah menerapkan sinyal supervisi berbasis klasifikasi ke setiap keluaran encoder. Misalnya, Clark et al. [2019] dalam Model ELECTRA, mengusulkan pelatihan encoder Transformer untuk mengidentifikasi apakah setiap token input identik dengan input asli atau telah diubah dalam beberapa cara. Langkah pertama dari metode ini adalah untuk menghasilkan urutan baru dari urutan token yang diberikan, di mana beberapa tokennya diubah. Untuk melakukan ini, model bahasa bertopeng kecil (sebut saja generator) diterapkan: kami secara acak menutupi beberapa token, dan melatih model ini untuk memprediksi token yang ditutupi. Untuk setiap pelatihan contohnya, model bahasa bertopeng ini mengeluarkan token di setiap posisi bertopeng, yang mungkin berbeda dari token asli. Pada saat yang sama, kami melatih encoder Transformer lain (sebut saja diskriminator) untuk menentukan apakah setiap token yang diprediksi sama dengan token asli atau diubah. Lebih khusus lagi, kami menggunakan generator untuk menghasilkan urutan di mana beberapa token diganti. Berikut adalah ilustrasinya.



Kemudian, kami menggunakan diskriminasi untuk memberi label masing-masing token ini sebagai asli atau terganti, sebagai berikut



Untuk pelatihan, generator dioptimalkan sebagai model bahasa bertopeng dengan kemungkinan maksimum estimasi, dan diskriminasi dioptimalkan sebagai pengklasifikasi menggunakan kerugian berbasis klasifikasi. ELECTRA, kerugian berbasis kemungkinan maksimum dan kerugian berbasis klasifikasi digabungkan untuk melatih generator dan diskriminasi secara bersamaan. Pendekatan alternatif adalah dengan menggunakan generatif jaringan adversarial (GAN), yaitu, generator dilatih untuk mengelabui diskriminasi, dan diskriminasi dilatih untuk membedakan keluaran generator dari distribusi sebenarnya. Namun, Pelatihan gaya GAN mempersulit tugas pelatihan dan lebih sulit untuk ditingkatkan. Namun demikian, setelah pelatihan selesai, generator dibuang, dan bagian pengkodean diskriminasi diterapkan sebagai model terlatih untuk tugas-tugas hilir.

1.2.3 Pra-pelatihan Encoder-Decoder

Dalam NLP, arsitektur encoder-decoder sering digunakan untuk memodelkan masalah urutan-ke-urutan, seperti penerjemahan mesin dan tanya jawab. Selain masalah urutan-ke-urutan yang umum dalam NLP ini, model encoder-decoder dapat diperluas untuk menangani banyak masalah lainnya. Ide sederhananya adalah menganggap teks sebagai masukan dan keluaran dari suatu masalah, sehingga kita dapat langsung menerapkan model encoder-decoder. Misalnya, jika diberikan sebuah teks, kita dapat meminta model untuk mengeluarkan teks yang menggambarkan sentimen dari teks masukan, seperti positif, negatif, dan netral.

Ide semacam itu memungkinkan kita mengembangkan sistem teks-ke-teks tunggal untuk mengatasi masalah NLP apa pun. Kita dapat merumuskan berbagai masalah ke dalam format teks-ke-teks yang sama. Pertama-tama kita melatih model encoder-decoder untuk memperoleh pengetahuan umum tentang bahasa melalui pengawasan mandiri. Model ini kemudian disempurnakan untuk tugas-tugas hilir tertentu menggunakan data teks-ke-teks yang ditargetkan.

1.2.3.1 Pra-pelatihan Encoder-Decoder Bertopeng

Dalam model T5 [Raffel et al. \[2020\]](#), banyak tugas berbeda dibingkai sebagai tugas teks-ke-teks yang sama. Setiap sampel di T5 mengikuti format

Teks Sumber → Teks Target

Di sini → memisahkan teks sumber, yang terdiri dari deskripsi tugas atau instruksi dan masukan yang diberikan ke sistem, dari teks target, yang merupakan respons terhadap tugas masukan. Sebagai contoh, pertimbangkan tugas menerjemahkan dari bahasa Mandarin ke bahasa Inggris. Contoh pelatihan dapat dinyatakan sebagai

[CLS] Terjemahkan dari Bahasa Mandarin ke Bahasa Inggris: 你是一个 Halo!

di mana [CLS] dan s adalah simbol awal di sisi sumber dan target, masing-masing⁵.

Demikian pula, kita dapat mengekspresikan tugas-tugas lain dengan cara yang sama. Misalnya

[CLS] Jawaban: Kapan Albert Einstein lahir?

→ s lahir pada tanggal 14 Maret 1879.

[CLS] Sederhanakan: profesor, yang telah menerbitkan banyak makalah di bidangnya, akan memberikan

kuliah tentang topik tersebut minggu depan. → Profesor

yang berpengalaman akan memberikan kuliah minggu depan.

[CLS] Skor terjemahan dari bahasa Inggris ke bahasa Mandarin. Bahasa Indonesia: ketika berada di Roma,

lakukan seperti yang dilakukan orang Romawi. Cina: 你是一个罗马人

→ dari 0,81

di mana instruksi disorot dalam warna abu-abu. Kasus yang menarik adalah bahwa dalam contoh terakhir kita

⁵Kita dapat menggunakan simbol awal yang sama untuk sekvensi yang berbeda. Di sini kita menggunakan simbol yang berbeda untuk membedakan sekvensi pada sisi encoder dan decoder.

membingkai ulang masalah penilaian sebagai masalah pembuatan teks. Tujuan kami adalah untuk menghasilkan teks yang mewakili angka 0,81, daripada mengeluarkannya sebagai nilai numerik.

Pendekatan yang dijelaskan di atas menyediakan kerangka kerja baru untuk pemahaman dan pembuatan bahasa universal. Baik instruksi tugas maupun masukan masalah diberikan ke sistem dalam bentuk teks. Sistem kemudian mengikuti instruksi untuk menyelesaikan tugas. Metode ini menyatukan berbagai masalah, dengan manfaat melatih satu model yang dapat melakukan banyak tugas secara bersamaan.

Secara umum, penyempurnaan diperlukan untuk mengadaptasi model yang telah dilatih sebelumnya ke tugas hilir tertentu. Dalam proses ini, seseorang dapat menggunakan berbagai cara untuk menginstruksikan model untuk tugas tersebut, seperti menggunakan nama tugas yang singkat sebagai awalan pada urutan masukan yang sebenarnya atau memberikan deskripsi terperinci tentang tugas tersebut. Karena instruksi tugas dinyatakan dalam bentuk teks dan terlibat sebagai bagian dari masukan, pengetahuan umum tentang instruksi dapat diperoleh melalui pembelajaran model pemahaman bahasa dalam fase pra-pelatihan. Ini dapat membantu memungkinkan pembelajaran tanpa percobaan. Misalnya, model yang telah dilatih sebelumnya dapat digeneralisasi untuk mengatasi masalah baru di mana instruksi tugas belum pernah ditemukan.

Ada beberapa metode pembelajaran mandiri yang ampuh untuk encoder atau decoder Transformer. Menerapkan metode ini untuk melatih model encoder-decoder secara awal relatif mudah. Salah satu pilihan yang umum adalah melatih model encoder-decoder sebagai model bahasa. Misalnya, encoder menerima prefiks sekuens, sementara decoder menghasilkan sekuens yang tersisa. Namun, ini berbeda dari pemodelan bahasa kausal standar, di mana seluruh sekuens dihasilkan secara autoregresif dari token pertama. Dalam kasus kami, encoder memproses prefiks sekaligus, dan kemudian decoder memprediksi token berikutnya dengan cara pemodelan bahasa kausal.

Lebih tepatnya, ini adalah masalah pemodelan bahasa awalan: suatu model bahasa memprediksi rangkaian berikutnya dengan awalan, yang berfungsi sebagai konteks untuk prediksi.

Perhatikan contoh berikut ini

[CLS] Anak anjing sedang bermain-main	ÿ di luar rumah.
Awalan	Urutan Berikutnya

Kita dapat langsung melatih model encoder-decoder menggunakan contoh seperti ini. Kemudian, encoder belajar memahami awalan, dan decoder belajar untuk terus menulis berdasarkan pemahaman ini.

Untuk pra-pelatihan berskala besar, mudah untuk membuat sejumlah besar contoh pelatihan dari teks yang tidak berlabel.

Perlu dicatat bahwa agar model encoder-decoder yang telah dilatih sebelumnya efektif dalam tugas multibahasa dan lintasbahasa, seperti penerjemahan mesin, model tersebut harus dilatih dengan data multibahasa. Hal ini biasanya mengharuskan kosakata mencakup token dari semua bahasa. Dengan demikian, model dapat mempelajari representasi bersama di berbagai bahasa, sehingga memungkinkan kemampuan dalam pemahaman dan pembuatan bahasa dalam konteks multibahasa dan lintasbahasa.

Pendekatan kedua untuk pra-pelatihan model encoder-decoder adalah pemodelan bahasa bertopeng. Dalam pendekatan ini, seperti yang dibahas dalam Bagian 1.2.2, token dalam suatu urutan secara acak diganti dengan simbol topeng, dan model tersebut kemudian dilatih untuk memprediksi token bertopeng ini berdasarkan seluruh urutan bertopeng.

Sebagai ilustrasi, pertimbangkan tugas menutupi dan merekonstruksi kalimat tersebut

Anak anjing sedang bermain-main di luar rumah .

Dengan menutupi dua token (misalnya, bermain-main dan), kita memiliki input dan output model bergaya BERT, sebagai berikut

[CLS] Anak-anak anjing itu [MASK] berada di luar rumah [MASK]
 ý dia — — — bermain-main_ itu — —

Di sini menunjukkan posisi bertopeng di mana kita tidak membuat prediksi token. Dengan memvariasikan persentase token dalam teks, pendekatan ini dapat digeneralisasi ke arah pelatihan gaya BERT atau pelatihan gaya pemodelan bahasa [Song et al., 2019]. Misalnya, jika kita menutupi semua token, maka model dilatih untuk menghasilkan seluruh urutan

[CLS] [MASKER] [MASKER] [MASKER] [MASKER] [MASKER] [MASKER] [MASKER] [MASKER]
 ý Anak -anak anjing sedang bermain-main di luar rumah.

Dalam kasus ini, kami melatih dekoder sebagai model bahasa.

Perhatikan bahwa, dalam konteks arsitektur encoder-decoder, kita dapat menggunakan encoder untuk membaca urutan yang ditutupi, dan menggunakan decoder untuk memprediksi urutan asli. Dengan tujuan ini, pada dasarnya kita memiliki autoencoder penghilang derau: encoder mengubah masukan yang rusak menjadi beberapa representasi tersembunyi, dan decoder merekonstruksi masukan yang tidak rusak dari representasi tersembunyi ini. Berikut adalah contoh masukan dan keluaran untuk pelatihan penghilang derau.

[CLS] Anak-anak anjing itu [MASK] berada di luar rumah [MASK].
 ý Anak -anak anjing sedang bermain-main di luar rumah.

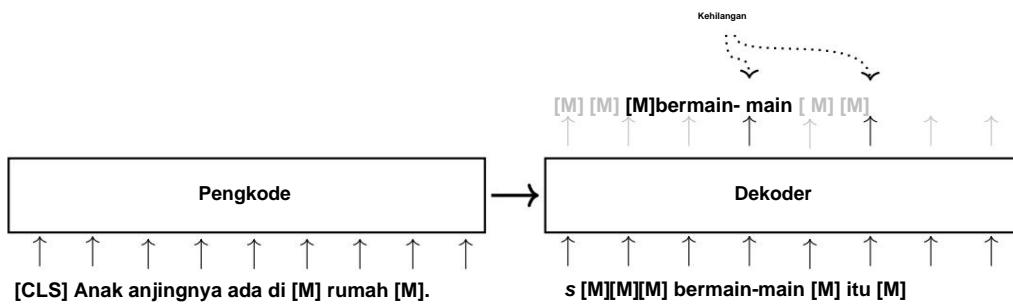
Dengan mempelajari cara memetakan dari urutan yang rusak ini ke bagian yang tidak rusak, model memperoleh kemampuan untuk memahami di sisi encoder dan menghasilkan di sisi decoder. Lihat Gambar 1.4 untuk ilustrasi tentang bagaimana model encoder-decoder dilatih dengan tujuan autoencoding gaya BERT dan denoising.

Karena kami memilih token secara acak untuk penyamaran, kami tentu dapat menyamarkan token yang berurutan [Joshi et al., 2020]. Berikut ini contohnya.

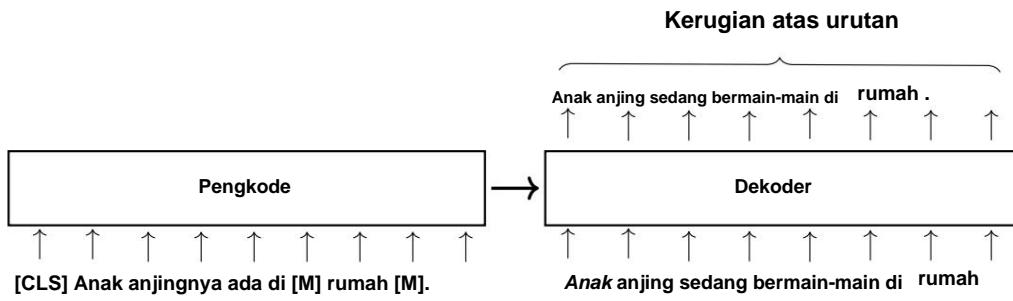
[CLS] Anak-anak anjingnya ada [MASK] di luar [MASK] [MASK] .
 ý Anak -anak anjing sedang bermain-main di luar rumah.

Cara lain untuk mempertimbangkan token bertopeng yang berurutan adalah dengan merepresentasikannya sebagai rentang. Di sini kami mengikuti karya Raffel et al. [2020], dan menggunakan [X], [Y], dan [Z] untuk menunjukkan token sentinel yang mencakup satu atau lebih token bertopeng yang berurutan. Dengan menggunakan notasi ini, kami dapat mengekspresikan kembali contoh pelatihan di atas sebagai

[CLS] Anak anjingnya ada [X] di luar [Y].
 ý s [X] bermain-main [Y] di rumah [Z]



(a) Melatih model encoder-decoder dengan pemodelan bahasa bertopeng gaya BERT



(b) Melatih model encoder-decoder dengan autoencoding denoising

Gbr. 1.4: Melatih model encoder-decoder menggunakan metode autoencoding gaya BERT dan denoising. Dalam kedua metode tersebut, input ke encoder adalah urutan token yang rusak di mana beberapa token ditutup dan diganti dengan [MASK] (atau singkatnya [M]). Dekoder memprediksi token yang ditutup ini, tetapi dengan cara yang berbeda. Dalam pelatihan gaya BERT, dekoder hanya perlu menghitung kerugian untuk token yang ditutup, sementara token yang tersisa dalam urutan tersebut dapat diperlakukan sebagai token [MASK]. Dalam autoencoding denoising, dekoder memprediksi urutan semua token secara autoregresif. Akibatnya, kerugian diperoleh dengan mengakumulasikan kerugian semua token ini, seperti dalam pemodelan bahasa standar.

Idenya adalah kita merepresentasikan urutan yang rusak sebagai urutan yang berisi slot pengganti. Tugas pelatihannya adalah mengisi slot ini dengan token yang benar menggunakan konteks di sekitarnya.

Keuntungan dari pendekatan ini adalah bahwa urutan yang digunakan dalam pelatihan akan lebih pendek, sehingga pelatihan menjadi lebih efisien. Perhatikan bahwa pemodelan bahasa bertopeng menyediakan kerangka kerja yang sangat umum untuk melatih model encoder-decoder. Berbagai pengaturan dapat disesuaikan untuk memiliki versi pelatihan yang berbeda, seperti mengubah persentase token yang ditutupi dan panjang maksimum token yang ditutupi. rentang.

1.2.3.2 Pelatihan Denoising

Jika kita melihat masalah pelatihan model encoder-decoder sebagai masalah pelatihan autoencoder penghilang derau, biasanya akan ada banyak metode berbeda untuk memperkenalkan kerusakan input dan merekonstruksi input. Misalnya, selain menutupi token secara acak, kita juga dapat mengubah beberapa di antaranya atau mengatur ulang urutannya.

Misalkan kita memiliki model encoder-decoder yang dapat memetakan urutan input x ke urutan output

urutan y

$$\begin{aligned} y &= \text{Dekode}\hat{y}(\text{Enkode}\hat{y}(x)) \\ &= \text{Model}\hat{y},\hat{y}(x) \end{aligned} \quad (1.15)$$

di mana \hat{y} dan \hat{y} masing-masing adalah parameter encoder dan decoder. Dalam denoising masalah autoencoding, kami menambahkan beberapa noise ke x untuk mendapatkan input x noise yang bising dan rusak. Dengan memasukkan x noise ke dalam encoder, kami berharap decoder mengeluarkan input asli. Pelatihan Tujuan dapat diartikan sebagai

$$(\hat{y}, \hat{y}) = \underset{\hat{y}, \hat{y}}{\arg \max} \quad \text{Kerugian}(\text{Model}\hat{y},\hat{y}(x\text{noise}), x) \quad (1.16)$$

Di sini fungsi kerugian $\text{Loss}(\text{Model}\hat{y},\hat{y}(x\text{noise}), x)$ mengevaluasi seberapa baik model $\text{Model}\hat{y},\hat{y}(x\text{noise})$ merekonstruksi input asli x . Kita dapat memilih kerugian entropi silang seperti biasa.

Ketika arsitektur model dan pendekatan pelatihan telah dikembangkan, masalah yang tersisa adalah korupsi input. [Lewis et al. \[2020\]](#), dalam model BART mereka, mengusulkan korupsi urutan masukan dalam beberapa cara berbeda.

- **Token Masking.** Ini adalah metode masking yang sama yang kami gunakan dalam mod-mod bahasa bertopeng. Token dalam urutan input dipilih secara acak dan ditutup.
- **Penghapusan Token.** Metode ini mirip dengan masking token. Namun, alih-alih mengganti token yang dipilih dengan simbol khusus [MASK], token ini dihapus dari urutan. Lihat contoh berikut untuk perbandingan token masking dan token metode penghapusan.

Asli (x): Anak-anak anjing sedang bermain-main di luar rumah

Token Masking (x noise): Anak anjing berada di luar rumah [MASK]

Penghapusan Token (x noise): Anak-anak anjing bermain - ~~main~~ di luar rumah

di mana token yang digarisbawahi pada urutan asli ditutupi atau dihapus.

- **Span Masking.** Span yang tidak tumpang tindih diambil sampelnya secara acak di seluruh urutan. Setiap rentang ditutupi oleh [MASK]. Kami juga mempertimbangkan rentang dengan panjang 0, dan, dalam kasus seperti itu, [MASK] hanya dimasukkan pada suatu posisi dalam urutan. Misalnya, kita dapat menggunakan span masking untuk merusak urutan di atas sebagai

Asli (x): 0 anak anjing sedang bermain-main di luar rumah

Span Masking (x noise): Anak anjing [MASK] adalah rumah [MASK]

Di sini rentang bermain-main di luar diganti dengan satu [MASK]. 0 menunjukkan rentang panjang-0, jadi kita memasukkan [MASK] antara The dan anak anjing. Penutupan rentang memperkenalkan tantangan prediksi baru di mana model perlu mengetahui berapa banyak token yang dihasilkan dari rentang waktu. Masalah ini sangat mirip dengan pemodelan kesuburan dalam penerjemahan mesin [\[Brown dan kawan-kawan, 1993\]](#).

Jika kita mempertimbangkan urutan yang terdiri dari beberapa kalimat, metode korupsi tambahan dapat diterapkan. Dalam model BART, ada dua metode tersebut.

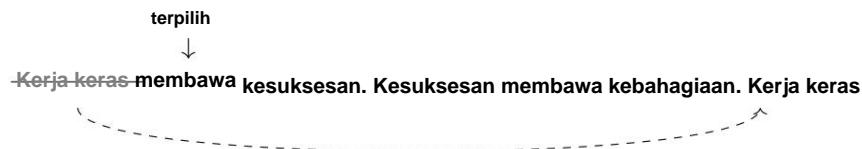
- **Penataan Ulang Kalimat.** Metode ini mengubah urutan kalimat secara acak sehingga model dapat belajar menyusun ulang kalimat dalam sebuah dokumen. Misalnya, perhatikan dua kalimat berurutan

Kerja keras membawa kesuksesan. Kesuksesan membawa kebahagiaan.

Kita dapat menyusun ulang kedua kalimat tersebut untuk mendapatkan urutan masukan yang rusak

Kesuksesan membawa kebahagiaan. Kerja keras membawa kesuksesan

- **Rotasi Dokumen.** Tujuan dari tugas ini adalah untuk mengidentifikasi token awal dari urutan tersebut. Pertama, sebuah token dipilih secara acak dari urutan tersebut. Kemudian, urutan tersebut diputar sehingga token yang dipilih adalah token pertama. Misalnya, misalkan kita memilih token yang mengarah dari urutan di atas. Urutan yang diputar adalah



di mana urutan Kerja keras sebelum memimpin ditambahkan pada akhir urutan.

Untuk pra-pelatihan, kita dapat menerapkan beberapa metode korupsi untuk mempelajari model yang kuat, misalnya, kita secara acak memilih salah satu dari mereka untuk setiap sampel pelatihan. Dalam praktiknya, hasil dari pelatihan awal encoder-decoder sangat bergantung pada metode korupsi input yang digunakan, dan oleh karena itu kami biasanya perlu memilih tujuan pelatihan yang tepat melalui eksperimen yang cermat.

1.2.4 Perbandingan Tugas Pra-Pelatihan

Sejauh ini, kita telah membahas sejumlah tugas pra-pelatihan. Karena tujuan pelatihan yang sama dapat berlaku untuk arsitektur yang berbeda (misalnya, menggunakan pemodelan bahasa bertopeng untuk encoder saja dan encoder-decoder pra-pelatihan), mengkategorikan tugas pra-pelatihan berdasarkan arsitektur model saja tampaknya tidak ideal. Sebaliknya, kami meringkas tugas-tugas ini berdasarkan tujuan pelatihan.

- **Pemodelan Bahasa.** Biasanya, pendekatan ini mengacu pada prosedur pembuatan urutan secara auto-regresif. Pada suatu waktu, ia memprediksi token berikutnya berdasarkan konteks sebelumnya.
- **Masked Language Modeling.** Masked Language Modeling termasuk dalam kerangka prediksi topeng umum. Ia secara acak menutupi token dalam suatu urutan dan memprediksi token ini menggunakan seluruh rangkaian yang bertopeng.

- **Permutasi Pemodelan Bahasa.** Permutasi pemodelan bahasa mengikuti ide yang sama dengan pemodelan bahasa bertopeng, tetapi mempertimbangkan urutan prediksi token (bertopeng). Ini menyusun ulang urutan input dan memprediksi token secara berurutan. Setiap prediksi didasarkan pada beberapa token konteks yang dipilih secara acak.
- **Pelatihan Diskriminatif.** Dalam pelatihan diskriminatif, sinyal supervisi dibuat dari tugas klasifikasi. Model untuk pra-pelatihan diintegrasikan ke dalam pengklasifikasi dan dilatih bersama dengan bagian pengklasifikasi yang tersisa untuk meningkatkan kinerja klasifikasinya.
- **Denoising Autoencoding.** Pendekatan ini diterapkan pada pra-pelatihan encoder-decoder model. Inputnya adalah urutan yang rusak dan model encoder-decoder dilatih untuk merekonstruksi urutan aslinya.

Tabel 1.1 mengilustrasikan metode-metode dan variannya menggunakan contoh-contoh. Penggunaan contoh-contoh ini tidak membedakan antara model-model, tetapi kami menandai arsitektur model di mana tugas pra-pelatihan dapat diterapkan. Dalam setiap contoh, input terdiri dari urutan token, dan outputnya berupa urutan token atau beberapa probabilitas. Untuk tugas pembangkitan, seperti bahasa pemodelan, superskrip digunakan untuk menunjukkan urutan pembangkitan pada sisi target. Jika superskrip dihilangkan, ini menunjukkan bahwa urutan keluaran dapat dihasilkan secara autoregresif atau secara bersamaan. Di sisi sumber, kami berasumsi bahwa urutan tersebut menjalani proses pengkodean Transformator standar, yang berarti bahwa setiap token dapat melihat seluruh urutan tersebut dengan perhatian penuh. Satu-satunya pengecualian adalah dalam pemodelan bahasa permutasi, di mana proses pembuatan autoregresif diimplementasikan dengan menetapkan topeng perhatian pada sisi encoder. Untuk menyederhanakan pembahasan, kami menghapus token *s* dari sisi target setiap contoh.

Meskipun tugas pra-pelatihan ini berbeda, adalah mungkin untuk membandingkannya dalam kerangka kerja dan pengaturan eksperimental yang sama [Dong et al., 2019; Raffel et al., 2020; Lewis et al., 2020]. Perhatikan bahwa kami tidak dapat mencantumkan semua tugas pra-pelatihan di sini karena jumlahnya banyak. Untuk diskusi lebih lanjut tentang tugas pra-pelatihan, pembaca yang tertarik dapat merujuk ke beberapa survei tentang topik ini [Qiu et al., 2020; Han dkk., 2021].

1.3 Contoh: BERT

Pada bagian ini, kami memperkenalkan model BERT, yang merupakan salah satu model paling populer dan banyak digunakan model pengkodean sekuens yang telah dilatih sebelumnya dalam NLP.

1.3.1 Model Standar

Model BERT standar, yang diusulkan dalam karya Devlin et al. [2019], adalah Transformer encoder dilatih menggunakan pemodelan bahasa bertopeng dan tugas prediksi kalimat berikutnya. Kerugian yang digunakan dalam pelatihan model ini merupakan penjumlahan kerugian dari kedua tugas.

$$\text{KerugianBERT} = \text{KerugianMLM} + \text{KerugianNSP} \quad (1.17)$$

Seperti yang biasa dilakukan dalam pelatihan jaringan saraf dalam, kami mengoptimalkan parameter model dengan meminimalkan kerugian ini. Untuk melakukan ini, sejumlah sampel pelatihan dikumpulkan. Selama pelatihan, sejumlah sampel

Metode	Enc	Des	ED	Masukan	Keluaran
LM Kausal		•	•		Anak kucing 1 , 2 , 3 , mengejar 4 , 5, bola . ⁷
Awalan LM		•	•	[C] Anak kucing itu	6 , mengejar 1, 2 , bola 3 ⁴
LM bertopeng	•			• [C] Anak kucing [M] mengejar [M] . • [C]	— — — bola —
Gaya MASS	•			Anak kucing [M] [M] [M] bola • [C] Anak	— — sedang mengejar — —
Gaya BERT	•			kucing [M] memainkan [M] .	— anak kucing mengejar bola —
LM yang dipermutasi	•			[C] Anak kucing itu mengejar bola	Anak kucing 5 mengejar 6 bola 4 2 ³
Kalimat Berikutnya Ramalan	•			[C] Anak kucing mengejar bola. Pr(IsNext representasi-dari-[C]) Burung memakan cacing.	
Kalimat Perbandingan	•			Mengkodekan kalimat sebagai <i>ha</i> dan kalimat lain sebagai <i>hb</i>	Skor (<i>ha</i> , <i>hb</i>)
Klasifikasi Token	•			[C] Anak kucing itu mengejar bola	Pr(- The) Pr(- anak kucing) ... Pr(- .)
Penataan Ulang Token				bola adalah anak kucing mengejar bola adalah anak kucing 1 mengejar bola adalah anak kucing 2 mengejar bola adalah anak kucing 3 mengejar bola adalah anak kucing 5 mengejar	. ⁷
Penghapusan Token			5 6	• [C] Anak kucing itu mengejar bola	Anak kucing 1 sedang mengejar 4 bola ⁷
Penutup Rentang				Anak kucing [M] adalah [M] .	Anak kucing 1 sedang mengejar 4 bola 5 6 • [C] ⁷
Penutup Sentinel				[C] Anak kucing [X] si [Y] • [C]	[X]1 mengejar2 bola3 [Y]4 5 • . ⁶
Kalimat Penataan ulang				Bola menggelinding dengan cepat. anak kucing mengejar bola	Anak kucing 1 sedang mengejar 4 bola 5. 6 bola 8. 9 menggelinding 10 menjauh 11 dengan cepat 12. ¹³
Dokumen			•	[C] mengejar bola	Bola menggelinding. 1 anak kucing, 2 mengejar, 3 bola, 4 bola, 6 bola, 8 ⁷
Rotasi				pergi dengan cepat. Anak kucing itu	bola, 9 menggelinding, 10 menjauh, 11 dengan cepat, 12. ¹³

Tabel 1.1: Perbandingan tugas pra-pelatihan, termasuk pemodelan bahasa, pemodelan bahasa bertopeng, pemodelan bahasa permutasi, pelatihan diskriminatif, dan pengodean otomatis penghilangan derau. [C] = [CLS], [M] = [MASK], [X], [Y] = token sentinel. Enc, Dec, dan ED menunjukkan apakah pendekatan tersebut dapat diterapkan pada model encoder-only, decoder-only, encoder-decoder, masing-masing. Untuk tugas pembangkitan, superskrip digunakan untuk merepresentasikan urutan token

sampel pelatihan dipilih secara acak dari koleksi ini pada suatu waktu, dan LossBERT diakumulasikan pada sampel pelatihan ini. Kemudian, parameter model diperbarui melalui penurunan gradien atau variannya. Proses ini diulang berkali-kali hingga beberapa kriteria penghentian terpenuhi, seperti saat kerugian konvergen.

1.3.1.1 Fungsi Kerugian

Secara umum, model BERT digunakan untuk merepresentasikan satu kalimat atau sepasang kalimat, dan dengan demikian dapat menangani berbagai masalah pemahaman bahasa hilir. Pada bagian ini kami berasumsi bahwa representasi input adalah urutan yang berisi dua kalimat SentA dan SentB, yang dinyatakan sebagai

[CLS] TerkirimA [SEP] TerkirimB [SEP]

Di sini kami mengikuti notasi dalam makalah BERT dan menggunakan [SEP] untuk menunjukkan pemisah.

Dengan urutan ini, kita dapat memperoleh LossMLM dan LossNSP secara terpisah. Untuk pemodelan bahasa bertopeng, kami memprediksi sebagian token dalam urutan tersebut. Biasanya, persentase tertentu dari

1.3 Contoh: BERT

23

token dipilih secara acak, misalnya, dalam model BERT standar, 15% token di setiap urutan dipilih. Kemudian urutan tersebut dimodifikasi dengan tiga cara

- Token Masking. 80% token yang dipilih akan ditutup dan diganti dengan simbol [MASKER]. Misalnya

Asli: [CLS] Sedang hujan. [SEP] Saya butuh payung. [SEP]
 Bertopeng: [CLS] Ini [MASK] . [SEP] Saya butuh [MASK] payung . [SEP]

di mana token yang dipilih digarisbawahi. Memprediksi token bertopeng membuat model mempelajari untuk merepresentasikan token dari konteks sekelilingnya.

- Penggantian Acak. 10% token yang dipilih diubah menjadi token acak. Untuk contoh,

Asli: [CLS] Sedang hujan. [SEP] Saya butuh payung. [SEP]
 Token Acak: [CLS] Sedang hujan. [SEP] Saya butuh **topi** . [SEP]

Ini membantu model belajar memulihkan token dari masukan yang berisik.

- Tidak berubah. 10% token yang dipilih tetap tidak berubah. Misalnya,

Asli: [CLS] Sedang hujan. [SEP] Saya butuh payung. [SEP]
 Token Tidak Berubah: [CLS] Sedang hujan. [SEP] **Saya** butuh payung. [SEP]

Ini bukan tugas prediksi yang sulit, tetapi dapat memandu model untuk menggunakan bukti yang lebih mudah untuk ramalan.

Misalkan $A(x)$ adalah himpunan posisi terpilih dari suatu deret token x , dan x^- adalah deret token termodifikasi. urutan x . Fungsi kerugian dari pemodelan bahasa bertopeng dapat didefinisikan sebagai

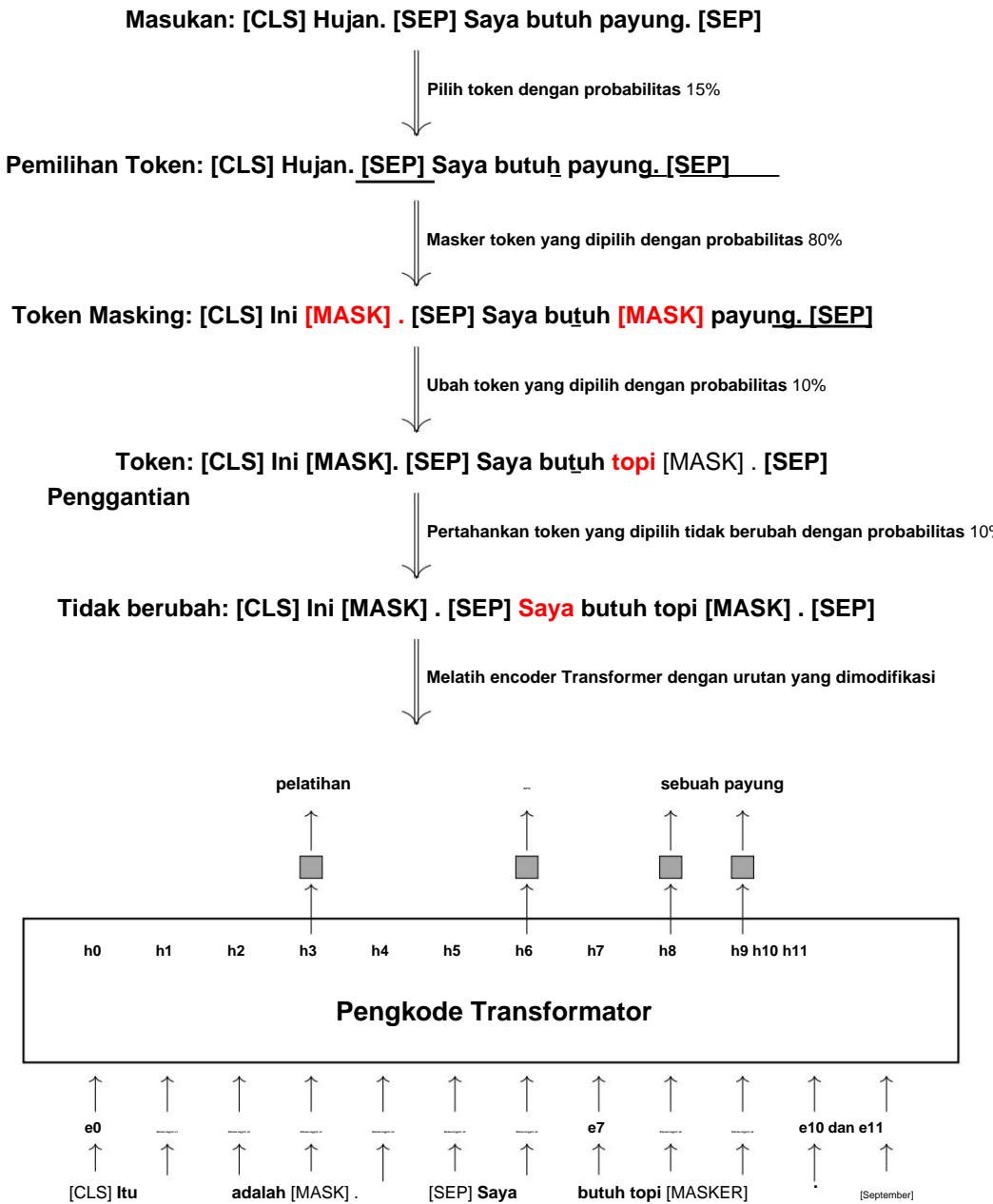
$$\text{KerugianMLM} = \hat{y} - \log \left(\frac{\text{probabilitas } \text{Pri}(x_i | x^-)}{\sum_{j \in A(x)} \text{probabilitas } \text{Pri}(x_j | x^-)} \right) \quad (1.18)$$

di mana $\text{Pri}(x_i | x^-)$ adalah probabilitas untuk memprediksi x_i pada posisi i jika diberikan x^- . Gambar 1.5 menunjukkan menjalankan contoh komputasi LossMLM.

Untuk prediksi kalimat berikutnya, kami mengikuti metode yang dijelaskan di Bagian 1.2.2.3. Setiap pelatihan sampel diklasifikasikan ke dalam set label {IsNext, NotNext}, misalnya,

Urutan: [CLS] Sedang hujan. [SEP] Saya butuh payung. [SEP]
 Label: Apakah Berikutnya

Urutan: [CLS] Kucing tidur di ambang jendela. [SEP] Apel tumbuh di pohon. [SEP]
 Label: TidakBerikutnya



Gambar 1.5: Contoh berjalan dari pemodelan bahasa bertopeng gaya BERT. Pertama, 15% token dipilih secara acak. Token yang dipilih ini kemudian diproses dengan salah satu dari tiga cara: diganti dengan token [MASK] (80% dari waktu), diganti dengan token acak (10% dari waktu), atau tetap tidak berubah (10% dari waktu). Model dilatih untuk memprediksi token-token yang dipilih ini berdasarkan urutan yang dimodifikasi. e_i mewakili penyematkan token pada posisi i . Gray kotak mewakili lapisan Softmax.

Vektor keluaran encoder untuk token pertama [CLS] dilihat sebagai representasi urutan, dilambangkan dengan h_{cls} (atau h_0). Sebuah pengklasifikasi dibangun di atas h_{cls} . Kemudian, kita dapat menghitung probabilitas label c diberikan h_{cls} , yaitu, $\Pr(c|h_{cls})$. Ada banyak fungsi kerugian yang dapat dipilih untuk klasifikasi masalah. Misalnya, dalam pelatihan kemungkinan maksimum, kita dapat mendefinisikan LossNSP sebagai

$$\text{KerugianNSP} = -\log \Pr(c_{gold}|h_{cls}) \quad (1.19)$$

1.3 Contoh: BERT

25

di mana c_{gold} adalah label yang benar untuk sampel ini.

1.3.1.2 Pengaturan Model

Seperti yang ditunjukkan pada Gambar 1.6, model BERT didasarkan pada arsitektur encoder Transformer standar.

Inputnya adalah urutan penyematan, yang masing-masing merupakan jumlah penyematan token, posisi, penanaman, dan penanaman segmen.

$$e = x + e_{pos} + e_{seg} \quad (1.20)$$

Baik penyematan token (x) dan penyematan posisi (e_{pos}) bersifat reguler, seperti pada Transformer model. Penanaman segmen (e_{seg}) adalah jenis penanaman baru yang menunjukkan apakah suatu token milik SentA atau SentB. Hal ini dapat diilustrasikan dengan contoh berikut.

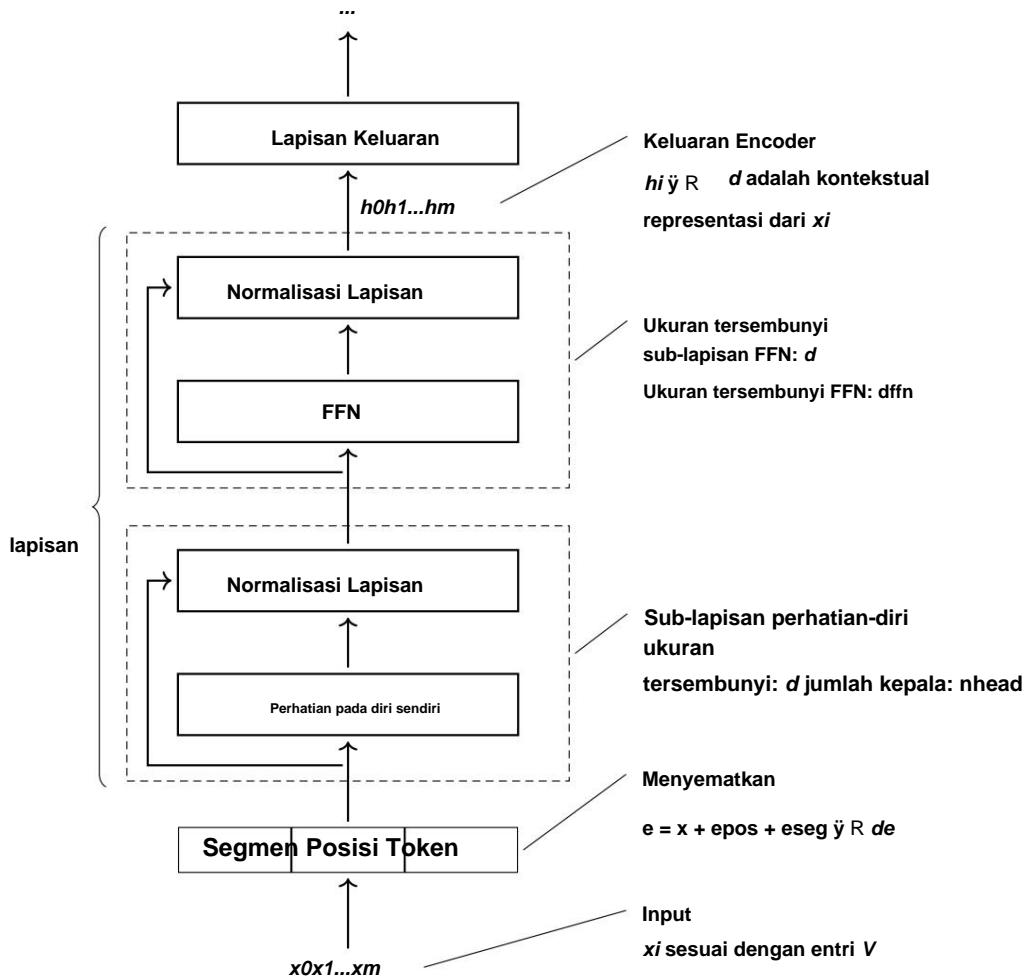
Token	[CLS]	Itu	sedang hujan	[SEP]	Aku butuh payung	.	[September]
	x0	x1x2	x3	ukuran x4x5x6x7x8		x9	x10 x11
epos		PE(0) PE(1) PE(2) PE(3) PE(4) PE(5) PE(6) PE(7) PE(8) PE(9) PE(10) PE(11)					
misalnya	eA	eA eA eA eA eB eB eB					Bahasa Indonesia: eB eB

Bagian utama dari model BERT adalah jaringan Transformer multi-layer. Lapisan Transformer terdiri dari sub-lapisan self-attention dan sub-lapisan FFN. Keduanya mengikuti post-norm arsitektur: $output = \text{LN}(F(\text{input}) + \text{input})$, di mana $F(\cdot)$ adalah fungsi inti dari sub-lapisan (baik model perhatian-diri atau FFN), dan $\text{LN}(\cdot)$ adalah unit normalisasi lapisan.

Biasanya, sejumlah lapisan Transformer ditumpuk untuk membentuk jaringan yang dalam. Pada setiap posisi urutannya, representasi keluarannya adalah vektor bernilai riil yang dihasilkan oleh lapisan terakhir jaringan.

Ada beberapa aspek yang dapat dipertimbangkan dalam mengembangkan model BERT.

- **Ukuran Kosakata ($|V|$).** Dalam Transformers, setiap token input direpresentasikan sebagai entri dalam Kosakata V . Kosakata yang besar dapat mencakup lebih banyak varian bentuk permukaan kata, tetapi mungkin menyebabkan peningkatan kebutuhan penyimpanan.
- **Ukuran Penyematan (de).** Setiap token direpresentasikan sebagai vektor bernilai riil yang tidak berdimensi. Seperti yang disajikan di atas, vektor ini adalah jumlah dari penyematan token, penyematan posisi, dan penanaman segmen, yang semuanya juga merupakan vektor bernilai riil yang tidak berdimensi.
- **Ukuran Tersembunyi (d).** Input dan output dari sub-lapisan memiliki dimensi d . Selain itu, sebagian besar dari status tersembunyi dari sub-lapisan adalah vektor d -dimensi. Secara umum, d dapat secara kasar dipandang sebagai lebar jaringan.
- **Jumlah Kepala (nhead).** Pada sub-lapisan self-attention, seseorang perlu menentukan jumlah jumlah kepala yang digunakan dalam perhatian diri multi-kepala. Semakin besar angka ini, semakin banyak sub-ruang perhatian dilakukan. Dalam sistem praktis, kita sering menetapkan nhead ≈ 4.
- **FFN Hidden Size (dffn).** Ukuran lapisan tersembunyi FFN yang digunakan dalam Transformer adalah biasanya lebih besar dari d . Misalnya, pengaturan tipikal adalah $dffn = 4d$. Untuk Transformer yang lebih besar, seperti model besar terkini, $dffn$ dapat ditetapkan ke nilai yang sangat besar.



Gambar 1.6: Arsitektur model BERT (Transformer encoder). Token input pertama kali direpresentasikan sebagai embedding, yang masing-masing merupakan jumlah dari embedding token, embedding posisional, dan embedding segmen yang sesuai. Kemudian, urutan penyematan diproses oleh tumpukan lapisan Transformer. Setiap lapisan dalam tumpukan ini mencakup sublapisan self-attention dan sublapisan FFN. Output dari model BERT adalah urutan vektor yang dihasilkan oleh lapisan Transformer akhir.

- **Kedalaman Model (L).** Menggunakan jaringan yang mendalam merupakan cara yang efektif untuk meningkatkan daya ekspresif Transformer. Untuk model BERT, L biasanya ditetapkan pada 12 atau 24. Namun, jaringan dengan kedalaman yang lebih besar juga layak dan dapat diterapkan untuk peningkatan lebih lanjut.

Pengaturan yang berbeda dari hiperparameter ini menghasilkan ukuran model yang berbeda. Ada dua model BERT yang umum digunakan.

- **BERTbase:** $d = 768$, $L = 12$, $nhead = 12$, jumlah total parameter = 110M.
- **BERTlarge:** $d = 1,024$, $L = 14$, $nhead = 16$, jumlah parameter = 340M.

Pelatihan model BERT mengikuti proses pelatihan standar Transformers. Pelatihan model yang lebih besar seperti BERTlarge memerlukan lebih banyak upaya dan waktu pelatihan. Ini adalah masalah umum untuk pra-pelatihan, terutama ketika model dilatih pada sejumlah besar data. Dalam praktiknya,

1.3 Contoh: BERT

27

sering kali ada pertimbangan efisiensi pelatihan. Misalnya, praktiknya adalah melatih model BERT terlebih dahulu pada urutan yang relatif pendek untuk sejumlah besar langkah pelatihan, lalu melanjutkan pelatihannya pada urutan dengan panjang penuh untuk langkah pelatihan yang tersisa.

1.3.2 Lebih Banyak Pelatihan dan Model yang Lebih Besar

BERT merupakan model penting dalam NLP, yang memicu banyak upaya berikutnya untuk menyempurnakannya. Salah satu arahnya adalah meningkatkan skala model itu sendiri, termasuk menambah data pelatihan dan mengembangkan model yang lebih besar.

RoBERTa, perluasan dari model BERT standar, adalah contoh dari upaya tersebut [Liu et al., 2019]. Model ini memperkenalkan dua peningkatan utama. Pertama, cukup dengan menggunakan lebih banyak data pelatihan dan lebih banyak komputasi dapat meningkatkan model BERT tanpa perlu mengubah arsitektur model. Kedua, menghilangkan kerugian NSP tidak menurunkan kinerja pada tugas hilir jika pelatihan ditingkatkan skalanya. Temuan ini menyarankan untuk mengeksplorasi arah umum pra-pelatihan: kita dapat terus meningkatkan pra-pelatihan dengan meningkatkan skalanya pada tugas pra-pelatihan yang sederhana.

Pendekatan kedua untuk meningkatkan model BERT adalah dengan meningkatkan jumlah parameter model. Misalnya, dalam karya He et al. [2021], model mirip BERT dengan 1,5 miliar parameter dibangun dengan meningkatkan kedalam model dan ukuran tersembunyi. Namun, peningkatan BERT dan berbagai model pra-terlatih lainnya menghadirkan tantangan baru dalam pelatihan, misalnya, pelatihan model yang sangat besar sering kali menjadi tidak stabil dan sulit untuk dikonvergensi. Hal ini membuat masalah menjadi lebih rumit, dan memerlukan pertimbangan cermat dari berbagai aspek, termasuk arsitektur model, komputasi paralel, inisialisasi parameter, dan sebagainya. Dalam contoh lain, Shoeybi et al. [2019] berhasil melatih model mirip BERT dengan 3,9 miliar parameter, di mana ratusan GPU digunakan untuk mengelola peningkatan tuntutan komputasi.

1.3.3 Model yang Lebih Efisien

Dibandingkan pendahulunya, BERT adalah model yang relatif besar pada saat diusulkan. Peningkatan ukuran model ini mengakibatkan kebutuhan memori yang lebih besar dan perlambatan kinerja sistem yang diakibatkannya. Mengembangkan model BERT yang lebih kecil dan lebih cepat merupakan bagian dari tantangan yang lebih luas dalam membangun Transformer yang efisien, yang telah dibahas secara luas dalam karya Tay et al. [2020] dan karya Xiao dan Zhu [2023]. Namun, pembahasan yang lebih mendalam tentang topik umum ini berada di luar cakupan pembahasan kita saat ini. Di sini, kami akan mempertimbangkan beberapa varian BERT yang efisien.

Beberapa alur penelitian menarik bagi peneliti NLP dalam mengembangkan model BERT yang efisien. Pertama, penelitian tentang penyulingan pengetahuan, seperti melatih model siswa dengan keluaran model guru yang terlatih dengan baik, menunjukkan bahwa model BERT yang lebih kecil dapat diperoleh dengan mentransfer pengetahuan dari model BERT yang lebih besar. Mengingat bahwa model BERT adalah jaringan multilapis dengan beberapa jenis lapisan yang berbeda, penyulingan pengetahuan dapat diterapkan pada berbagai tingkat representasi. Misalnya, selain menyaring pengetahuan dari lapisan keluaran, juga memungkinkan untuk menggabungkan kerugian pelatihan yang mengukur perbedaan keluaran lapisan tersembunyi antara model guru dan model siswa [Sun et al., 2020; Jiao et al., 2020]. Memang, penyulingan pengetahuan telah menjadi salah satu teknik yang paling banyak digunakan untuk mempelajari model kecil yang telah dilatih sebelumnya.

Kedua, metode kompresi model konvensional dapat langsung diterapkan untuk mengompresi model BERT. Salah satu pendekatan umum adalah menggunakan metode pemangkas serbaguna untuk memangkas jaringan pengkodean Transformator [Gale et al., 2019]. Hal ini umumnya melibatkan penghapusan seluruh lapisan

[Fan et al., 2019] atau persentase parameter tertentu dalam jaringan [Sanh et al., 2020; Chen et al., 2020]. Pemangkas juga berlaku untuk model perhatian multi-head. Misalnya, Michel et al. [2019] menunjukkan bahwa menghilangkan beberapa head tidak secara signifikan menurunkan kinerja model BERT, tetapi mempercepat inferensi model ini. Pendekatan lain untuk mengompresi model BERT adalah kuantisasi [Shen et al., 2020]. Dengan merepresentasikan parameter model sebagai angka presisi rendah, model dapat dikompresi secara signifikan. Meskipun metode ini tidak khusus untuk model BERT, metode ini terbukti efektif untuk arsitektur berbasis Transformer yang besar.

Ketiga, mengingat model BERT adalah jaringan yang relatif dalam dan besar, rangkaian penelitian lain menggunakan jaringan dinamis untuk mengadaptasi model ini demi inferensi yang efisien. Sebuah ide dalam paradigma ini adalah memilih lapisan secara dinamis untuk memproses token, misalnya, dalam model adaptif kedalaman, kita keluar pada kedalaman optimal tertentu dan dengan demikian melewati lapisan lainnya dalam tumpukan lapisan [Xin et al., 2020; Zhou et al., 2020]. Demikian pula, kita dapat mengembangkan model adaptif panjang di mana panjang urutan masukan disesuaikan secara dinamis. Misalnya, kita dapat melewati beberapa token dalam urutan masukan sehingga model dapat mengurangi beban komputasi pada token yang kurang penting, meningkatkan efisiensi secara keseluruhan.

Keempat, dimungkinkan juga untuk berbagi parameter lintas lapisan guna mengurangi ukuran model BERT. Cara sederhana untuk melakukan ini adalah dengan membagikan parameter seluruh lapisan Transformer di seluruh tumpukan lapisan [Dehghani et al., 2018; Lan et al., 2020]. Selain mengurangi jumlah parameter, hal ini memungkinkan penggunaan kembali lapisan yang sama dalam jaringan Transformer multilapis, sehingga menghemat jejak memori pada waktu pengujian.

1.3.4 Model Multibahasa

Model BERT awal difokuskan pada bahasa Inggris. Segera setelah model ini diusulkan, model ini diperluas ke banyak bahasa. Salah satu cara sederhana untuk melakukannya adalah dengan mengembangkan model terpisah untuk setiap bahasa. Pendekatan lain, yang menjadi lebih populer dalam karya terbaru pada model bahasa besar, adalah melatih model multibahasa secara langsung pada data dari semua bahasa. Sebagai tanggapan, model BERT multibahasa (mBERT) dikembangkan dengan melatihnya pada 104 bahasa⁶. Teks dari Perbedaan utama dari model BERT monolingual adalah bahwa model mBERT menggunakan kosakata yang lebih besar untuk mencakup token dari berbagai bahasa. Hasilnya, representasi token dari berbagai bahasa dipetakan ke dalam ruang yang sama, yang memungkinkan pembagian pengetahuan lintas bahasa melalui model representasi universal ini.

Salah satu aplikasi penting dari model pra-latihan multibahasa adalah pembelajaran lintas bahasa. Dalam konteks lintas bahasa, kita mempelajari model pada tugas dalam satu bahasa, dan menerapkannya pada tugas yang sama dalam bahasa lain. Dalam klasifikasi teks lintas bahasa, misalnya, kita menyempurnakan model pra-latihan multibahasa pada dokumen beranotasi bahasa Inggris. Kemudian, kita menggunakan model yang disempurnakan untuk mengklasifikasikan dokumen bahasa Mandarin.

Peningkatan pada model pra-latihan multibahasa seperti mBERT adalah dengan memperkenalkan data dwibahasa ke dalam pra-pelatihan. Daripada melatih hanya pada data monolingual dari berbagai bahasa, pelatihan dwibahasa secara eksplisit memodelkan hubungan antara token dalam dua bahasa. Model yang dihasilkan akan memiliki kemampuan transfer lintas bahasa bawaan, dan dengan demikian dapat dengan mudah diadaptasi ke berbagai bahasa. Lample dan Conneau [2019] mengusulkan pendekatan untuk pra-pelatihan model bahasa lintas bahasa (XLM). Dalam karya mereka, model bahasa lintas bahasa dapat dilatih baik dalam pemodelan bahasa kausal maupun pemodelan bahasa bertopeng. Untuk pemodelan bah-

⁶<https://github.com/google-research/bert/>

pra-pelatihan, model diperlakukan sebagai encoder. Tujuan pelatihan sama dengan BERT: kita memaksimalkan kemungkinan beberapa token yang dipilih secara acak yang ditutupi, diganti dengan token acak, atau tetap tidak berubah dalam input. Jika kita mempertimbangkan data bilingual dalam pra-pelatihan, kami mengambil sampel sepasang kalimat yang selaras setiap waktu. Kemudian, kedua kalimat tersebut dikemas bersama untuk membentuk satu rangkaian tunggal yang digunakan untuk pelatihan. Misalnya, perhatikan pasangan kalimat bahasa Inggris-bahasa Mandarin

ÿÿ ÿ ÿÿÿÿ ÿ Paus adalah mamalia .

Kita dapat mengemasnya untuk mendapatkan urutan seperti ini

[CLS] ÿÿ ÿ ÿÿ [SEP] Paus adalah mamalia. [SEP]

Kami kemudian memilih persentase token tertentu dan menggantinya dengan [MASK].

[CLS] [MASK] ÿ [MASK] ÿÿ [SEP] Paus [MASK] [MASK] . [SEP]

Tujuan dari pra-pelatihan adalah untuk memaksimalkan produk dari probabilitas token bertopeng yang diberikan urutan di atas. Dengan melakukan pelatihan dengan cara ini, model dapat belajar untuk mewakili keduanya Urutan bahasa Inggris dan Cina, serta untuk menangkap korespondensi antara token di dua bahasa. Misalnya, memprediksi token Cina ÿÿ mungkin memerlukan informasi dari token bahasa Inggris Whales. Menyelaraskan representasi kedua bahasa pada dasarnya mengubah model menjadi model “terjemahan”. Jadi tujuan pelatihan ini juga disebut pemodelan bahasa terjemahan. Gambar 1.7 menunjukkan ilustrasi pendekatan ini.

Manfaat model pra-latihan multibahasa adalah kemampuan bawaannya dalam menangani peralihan kode. Dalam NLP dan linguistik, peralihan kode mengacu pada peralihan antarbahasa dalam sebuah teks. Misalnya, berikut ini adalah teks bahasa campuran yang mengandung bahasa Mandarin dan Inggris:

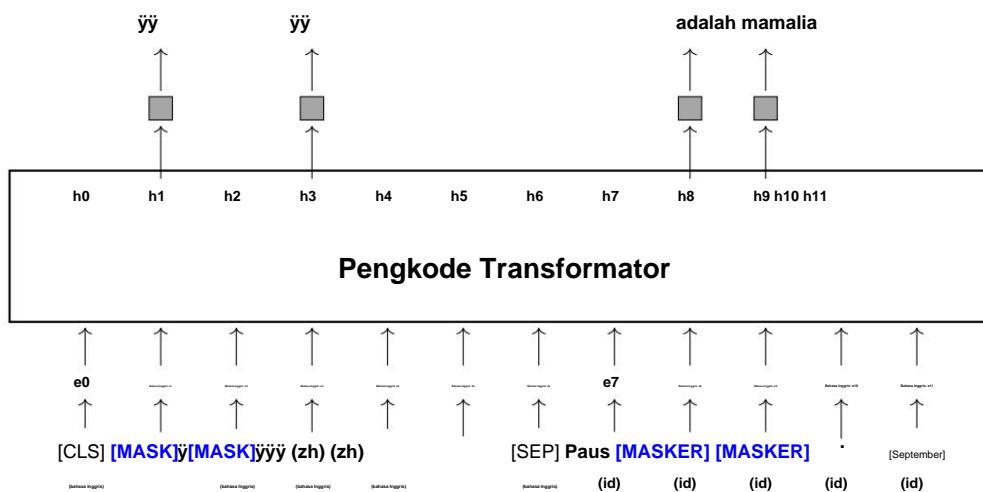
ÿÿ ÿÿ ÿÿ ÿ mendaki ÿ

ÿ ÿ ÿÿ ÿ ÿÿ

(Kami berencana pergi hiking akhir pekan ini, apakah kamu mau ikut?)

Untuk model pra-latihan multibahasa, kita tidak perlu mengidentifikasi apakah token tersebut berbahasa Mandarin atau Bahasa Inggris. Sebaliknya, setiap token hanyalah entri kosakata bersama. Hal ini dapat dibayangkan sebagai menciptakan bahasa “baru” yang mencakup semua bahasa yang ingin kita proses.

Hasil dari pra-pelatihan multibahasa dipengaruhi oleh beberapa faktor. Mengingat model arsitekturnya tetap, seseorang perlu menentukan ukuran kosakata bersama, jumlah (atau persentase) sampel dalam setiap bahasa, ukuran model, dan sebagainya. Conneau et al. [2020] menunjukkan beberapa masalah menarik mengenai pra-pelatihan multibahasa skala besar untuk XLM-like model. Pertama, seiring dengan meningkatnya jumlah bahasa yang didukung, model yang lebih besar diperlukan untuk menangani Kedua, kosakata bersama yang lebih banyak membantu dalam pemodelan peningkatan keragaman bahasa. Ketiga, bahasa dengan sumber daya terbatas lebih mudah memperoleh manfaat dari transfer lintas bahasa. dari bahasa-bahasa yang membutuhkan sumber daya tinggi, terutama ketika bahasa-bahasa yang membutuhkan sumber daya tinggi serupa terlibat di dalamnya pra-pelatihan. Namun, gangguan dapat terjadi jika model dilatih dalam jangka waktu yang lama,



Gambar 1.7: Ilustrasi pemodelan bahasa terjemahan. Untuk memudahkan pemahaman, kami berikan contoh sederhana di mana semua token yang dipilih ditutupi. Model dilatih untuk memprediksi token yang ditutupi ini. Saat urutan berisi token dalam dua bahasa, memprediksi token dalam satu bahasa memungkinkan akses ke token dalam bahasa lain, sehingga memungkinkan pemodelan lintas bahasa. Dalam karya Lample dan Conneau [2019], penyematan input (yaitu, e_i) adalah jumlah dari penyematan token, penyematan posisi, dan penyematan bahasa. Ini mengharuskan setiap token ditetapkan dengan label bahasa. Dengan demikian kita dapat membedakan token dalam bahasa yang berbeda. Dalam pra-pelatihan multibahasa, khususnya dalam pekerjaan yang menggunakan kosakata bersama, tidak perlu menentukan bahasa yang digunakan token. Penggunaan penyematan bahasa pada gilirannya membuat sulit untuk menangani peralihan kode. Oleh karena itu, kami berasumsi di sini bahwa semua token representasi bersifat independen terhadap bahasa.

artinya kinerja keseluruhan model yang telah dilatih sebelumnya mulai menurun pada titik tertentu selama pra-pelatihan. Jadi, dalam sistem praktis, seseorang mungkin perlu menghentikan pra-pelatihan lebih awal untuk mencegah gangguan.

1.4 Menerapkan Model BERT

Setelah model BERT dilatih terlebih dahulu, model tersebut dapat digunakan untuk memecahkan masalah NLP. Namun model BERT tidak langsung siap untuk melakukan tugas hilir tertentu. Secara umum, pekerjaan penyempurnaan tambahan diperlukan untuk membuat mereka beradaptasi. Sebagai langkah pertama, kita memerlukan prediktor untuk menyelaraskan keluaran model dengan permasalahan yang diinginkan. Misalkan $BERT^{\hat{y}}(\cdot)$ adalah model BERT dengan parameter yang telah dilatih sebelumnya \hat{y} , dan $Predict_{\hat{y}}(\cdot)$ adalah jaringan prediksi dengan parameter \hat{y} . Dengan mengintegrasikan jaringan prediksi dengan output model BERT, kami mengembangkan model untuk mengatasi tugas hilir. Model ini dapat dinyatakan sebagai

$$y = Predict_{\hat{y}}(BERT^{\hat{y}}(x)) \quad (1.21)$$

di mana x adalah input dan y adalah output yang sesuai dengan masalah. Misalnya, dalam klasifikasi masalah, model mengeluarkan distribusi probabilitas atas label.

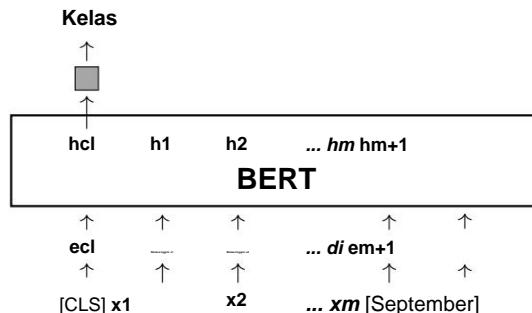
Kemudian, kami mengumpulkan sekumpulan sampel berlabel D , dan menyempurnakan modelnya dengan

$$(\hat{y}, \tilde{y}) = \arg \min_{\hat{y}, \tilde{y}} \text{Rugi}(y, \hat{y} + (x, y) D) \quad (1.22)$$

di mana (x, y_{gold}) mewakili tupel input dan output yang sesuai. Notasi ini persamaan tampaknya agak rumit, tetapi proses pelatihan/penyetelan adalah standar. Kami mengoptimalkan model dengan meminimalkan kerugian pada sampel penyetelan. Hasilnya adalah parameter yang dioptimalkan \tilde{y} dan \tilde{y}' . Optimasi dimulai dengan parameter yang telah dilatih sebelumnya \tilde{y} . Di sini kita menggunakan \tilde{y}^+ untuk menunjukkan bahwa parameter diinisialisasi dengan \tilde{y} , dan digunakan untuk menunjukkan keluaran model yang dihitung menggunakan parameter \tilde{y} dan \tilde{y}^+ .

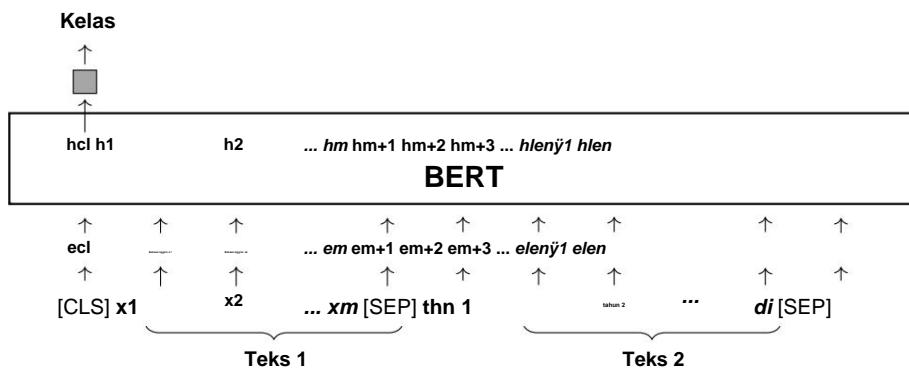
Dengan parameter fine-tuned \tilde{y} dan \tilde{y}' , kita dapat menerapkan model $\text{Predict}(\cdot)$ (BERT) ke model baru data tugas yang sama yang modelnya telah disempurnakan. Bentuk tugas hilir menentukan format input dan output model, serta arsitektur prediksi jaringan. Berikut ini kami cantumkan beberapa tugas yang umumnya cocok untuk model BERT.

- **Klasifikasi (Teks Tunggal).** Salah satu aplikasi model BERT yang paling banyak digunakan adalah klasifikasi teks. Dalam tugas ini, model BERT menerima serangkaian token dan mengodekannya itu sebagai urutan vektor. Vektor keluaran pertama h_{cls} (atau h_0) biasanya digunakan sebagai representasi seluruh teks. Jaringan prediksi mengambil h_{cls} sebagai input untuk menghasilkan distribusi label. Biarkan $[CLS]x_1x_2\dots x_m$ menjadi teks input. Lihat di bawah untuk ilustrasi Klasifikasi teks berbasis BERT.



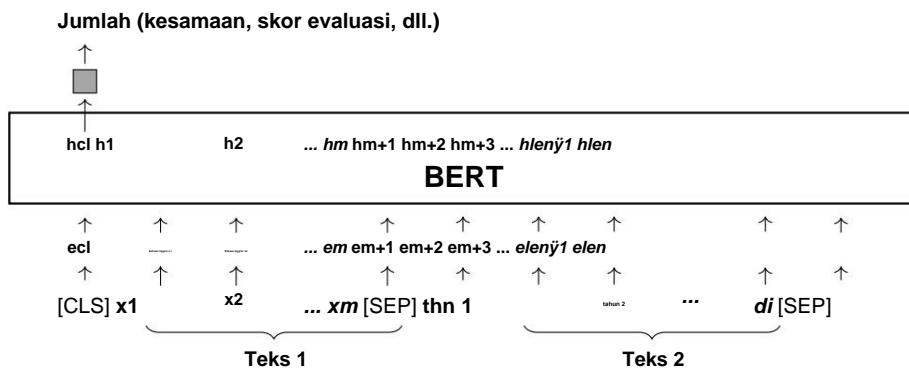
Di sini kotak abu-abu menunjukkan jaringan prediksi. Banyak masalah NLP dapat dikategorikan sebagai tugas klasifikasi teks, dan ada beberapa tolok ukur klasifikasi teks untuk mengevaluasi model yang telah dilatih sebelumnya. Misalnya, kita dapat mengklasifikasikan teks berdasarkan ketepatan tata bahasa (gramatikalitas) atau nada emosional (sentimen) [Socher et al., 2013; Warstadt et al., 2019]. Perhatikan bahwa jaringan prediksi bisa berupa model klasifikasi apa pun, seperti deep jaringan saraf atau model klasifikasi yang lebih tradisional. Seluruh model kemudian dapat dilatih atau disempurnakan dengan cara model klasifikasi standar. Misalnya, jaringan prediksi dapat berupa lapisan Softmax dan parameter model dapat dioptimalkan dengan memaksimalkan kemungkinan label yang benar.

- **Klasifikasi (Sepasang Teks).** Klasifikasi juga dapat dilakukan pada sepasang teks. Misalkan kita memiliki dua teks, $x_1\dots x_m$ dan $y_1\dots y_n$. Kita dapat menggabungkan teks-teks ini untuk membentuk satu teks tunggal. urutan dengan panjang len . Kemudian, kami memprediksi label untuk urutan teks gabungan ini berdasarkan pada vektor h_{cls} , sebagai berikut



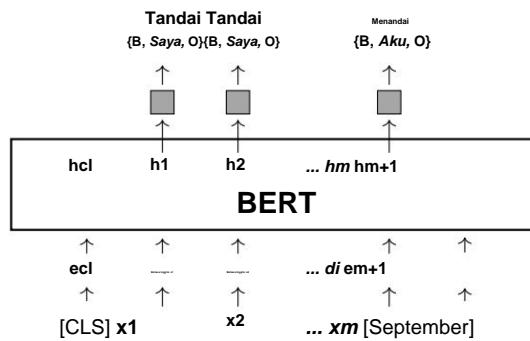
di mana $len = n + m + 2$. Klasifikasi pasangan teks mencakup beberapa masalah, termasuk penilaian kesetaraan semantik (menentukan apakah dua teks setara secara semantik) [Dolan dan Brockett, 2005], penilaian implikasi teks (menentukan apakah suatu hipotesis dapat disimpulkan secara logis atau disimpulkan dari suatu premis) [Bentivogli dan Giampiccolo, 2011; Williams et al., 2018], inferensi akal sehat yang mendasar (menentukan apakah suatu peristiwa kemungkinan terjadi mengingat konteksnnya) [Zellers et al., 2018], dan inferensi tanya jawab (menentukan apakah jawaban sesuai dengan pertanyaan yang diberikan).

- **Regresi.** Daripada membuat distribusi label, kita dapat memiliki jaringan prediksi menghasilkan skor bernilai riil. Misalnya, dengan menambahkan lapisan Sigmoid ke jaringan prediksi, sistem dapat digunakan untuk menghitung kesamaan antara dua kalimat yang diberikan. Arsitekturnya sama dengan sistem klasifikasi berbasis BERT, hanya saja perubahan lapisan keluaran.



Untuk pelatihan atau penyempurnaan, kita dapat meminimalkan kerugian regresi keluaran model sebagai biasa.

- **Pelabelan Urutan.** Pelabelan urutan adalah pendekatan pembelajaran mesin yang berlaku untuk berbagai berbagai masalah NLP. Pendekatan ini memberikan label pada setiap token dalam urutan input, dan beberapa anotasi linguistik kemudian dapat diturunkan dari urutan label ini. Contoh pelabelan urutan dalam NLP adalah penandaan part-of-speech (POS). Kami memberi label pada setiap kata dalam kalimat dengan tag POS yang sesuai. Contoh lain disebut pengenalan entitas (NER) di mana kita memberi label setiap kata dengan tag NER, dan entitas bernama diidentifikasi menggunakan tag ini. Lihat di bawah untuk ilustrasi arsitektur model untuk NER.

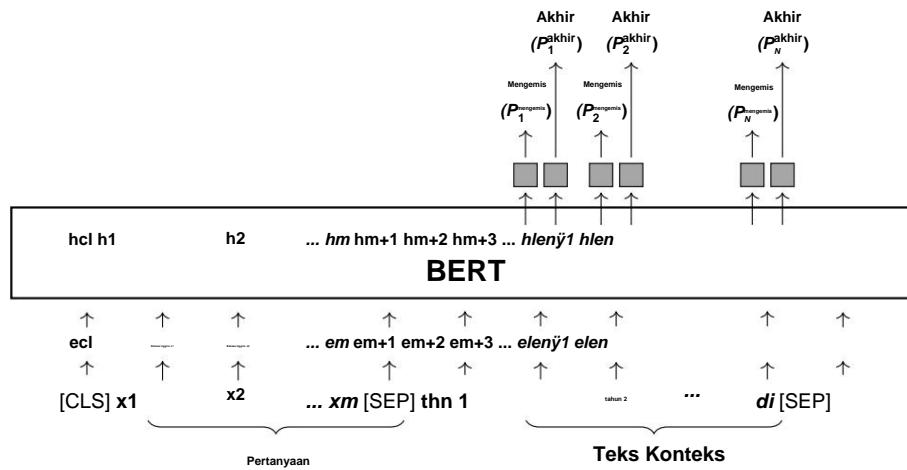


Di sini $\{B, I, O\}$ adalah set tag NER. Misalkan, B-ORG berarti awal dari organisasi, I-ORG berarti kata tersebut ada di dalam organisasi, dan O berarti kata tersebut tidak ada di dalam organisasi. tidak termasuk dalam entitas bernama mana pun. Model NER ini dapat menghasilkan distribusi atas kumpulan tag pada setiap posisi, dilambangkan sebagai p pelatihan atau penyempurnaan model dapat dilakukan atas distribusi ini $\{p_1, \dots, p_m\}$. Misalkan $p_i(\text{tag}_i)$ adalah probabilitas tag yang benar pada posisi i . Kerugian pelatihan dapat didefinisikan sebagai kemungkinan negatif

$$\text{Kerugian} = \ddot{y} \cdot \frac{1}{M} \sum_{i=1}^M \log p_i(\text{tag}_i) \quad (1.23)$$

Menemukan urutan label terbaik dengan model NER yang terlatih merupakan masalah yang dipelajari dengan baik dalam NLP. Hal ini sering dicapai melalui pemrograman dinamis, yang, dalam konteks pencarian jalur atas kisi, memiliki kompleksitas linier [Huang, 2009].

- Prediksi Rentang. Beberapa tugas NLP memerlukan prediksi rentang dalam sebuah teks. Contoh umum adalah pemahaman bacaan. Dalam tugas ini, kita diberikan query $x_1 \dots x_m$ dan teks konteks $y_1 \dots y_n$. Tujuannya adalah untuk mengidentifikasi rentang kontinu di $y_1 \dots y_n$ yang paling tepat untuk menjawab pertanyaan tersebut. Masalah ini dapat dibingkai sebagai tugas pelabelan urutan di mana kita memprediksi label untuk setiap y_j untuk menunjukkan awal atau akhir rentang. Mengikuti Seo et al. [2017], kami menambahkan dua jaringan di atas output BERT untuk y_j : satu untuk menghasilkan probabilitas y_j menjadi awal rentang (dilambangkan dengan $p_{j \text{ awal}}^{menginis}$), dan satu untuk menghasilkan probabilitas dari y_j menjadi akhir rentang (dilambangkan dengan $p_{j \text{ akhir}}^{menginis}$). Arsitektur model yang dihasilkan adalah ditunjukkan sebagai berikut



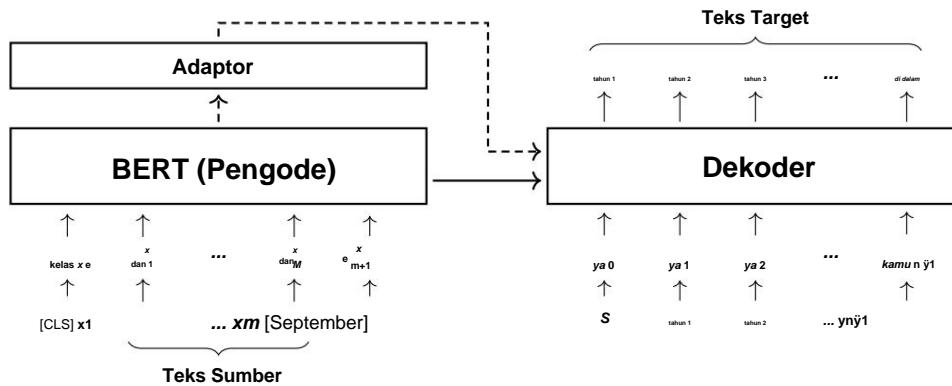
Kami mengemas kueri dan teks konteks bersama-sama untuk memperoleh urutan masukan. Jaringan awal prediksi hanya diterapkan pada keluaran untuk teks konteks, menghasilkan probabilitas p_j dan p_{j+1} dua model di seluruh teks akhir pada setiap posisi. Kerugian dapat dihitung dengan menjumlahkan log kemungkinan konteks.

$$\text{Kerugian} = \hat{y} - \frac{1}{n_j} \sum_{j=1}^N \text{mohor}_j \log p_j + \log p_{j+1}^{\text{akhir}} \quad (1.24)$$

Pada saat pengujian, kami mencari rentang terbaik dengan

$$(\hat{j}_1, \hat{j}_2) = \arg \max_{\hat{j}_1, \hat{j}_2} \text{mohor}_{\hat{j}_1, \hat{j}_2} \log p_j + \log p_{j+1}^{\text{akhir}} \quad (1.25)$$

- Pengodean untuk Model Encoder-decoder. Meskipun fokus kita di bagian ini terutama pada masalah pemahaman bahasa, perlu dicatat bahwa model BERT dapat diterapkan pada berbagai tugas NLP yang lebih luas. Faktanya, model BERT dapat digunakan dalam semua skenario saat kita perlu mengodekan sepotong teks. Satu aplikasi yang belum kita sebutkan adalah pembuatan teks yang mencakup berbagai tugas seperti penerjemahan mesin, peringkasan, menjawab pertanyaan, dan pembuatan dialog. Tugas-tugas ini dapat diformulasikan sebagai masalah urutan-ke-urutan: kita menggunakan encoder untuk merepresentasikan teks sumber, dan decoder untuk menghasilkan teks target yang sesuai. Metode langsung untuk menerapkan model BERT adalah dengan menganggapnya sebagai encoder. Sebelum melakukan fine-tuning, kita dapat menginisialisasi parameter encoder dengan parameter dari model BERT yang telah dilatih sebelumnya. Kemudian, model encoder-decoder dapat disempurnakan pada pasangan teks seperti biasa. Berikut ini menunjukkan arsitektur sistem penerjemahan mesin saraf di mana model BERT diterapkan di sisi sumber.



Di sini $x_1 \dots x_m$ menunjukkan urutan sumber, $y_1 \dots y_n$ menunjukkan urutan target, e menunjukkan urutan penyisipan $x_1 \dots x_m$, dan e menunjukkan urutan $y_1 \dots y_n$. Adaptor, yang bersifat opsional, memetakan keluaran model BERT ke bentuk yang lebih sesuai dengan dekoder.

Penyetelan halus model BERT merupakan masalah rekayasa rumit yang dipengaruhi oleh banyak faktor, seperti jumlah data penyetelan halus, ukuran model, dan pengoptimalan yang digunakan dalam penyetelan halus. Secara umum, kami ingin menyempurnakan model-model ini secara memadai sehingga dapat bekerja dengan baik dalam tugas-tugas hilir. Namun, menyempurnakan model BERT untuk tugas-tugas tertentu dapat menyebabkan overfitting,

yang pada gilirannya mengurangi kemampuan mereka untuk menggeneralisasi ke tugas-tugas lain. Misalnya, anggaplah kita memiliki Model BERT yang berkinerja baik pada tugas tertentu. Jika kita kemudian menyempurnakannya untuk tugas baru, ini dapat menurunkan kinerjanya pada tugas aslinya. Masalah ini terkait dengan bencana masalah lupa dalam pelatihan berkelanjutan, di mana jaringan saraf melupakan informasi yang dipelajari sebelumnya ketika diperbarui pada sampel baru. Dalam aplikasi praktis, cara umum untuk meringankan lupa yang fatal adalah menambahkan beberapa data lama ke dalam fine-tuning dan melatih model dengan lebih banyak data yang beragam. Selain itu, seseorang dapat menggunakan metode yang dikhususkan untuk mengatasi lupa yang sangat parah, seperti pemutaran pengalaman [Rønning et al., 2019] dan konsolidasi berat elastis [Kirkpatrick et al., 2017]. pembaca yang tertarik dapat merujuk ke beberapa survei untuk diskusi lebih rinci tentang masalah ini secara berkelanjutan pembelajaran [Parisi et al., 2019; Wang et al., 2023a;e].

1.5 Ringkasan

Dalam bab ini, kami telah membahas gagasan umum tentang pra-pelatihan dalam NLP. Secara khusus, kami telah membahas pra-pelatihan yang diawasi sendiri dan penerapannya pada arsitektur encode-only, decoder-only, dan encoder-decoder. Selain itu, kami telah menyajikan dan membandingkan berbagai tugas pra-pelatihan. untuk arsitektur ini. Sebagai contoh, BERT digunakan untuk mengilustrasikan bagaimana model sekvens dilatih terlebih dahulu melalui pemodelan bahasa bertopeng dan diterapkan ke berbagai tugas hilir.

Beberapa tahun terakhir telah menunjukkan kemajuan yang luar biasa dalam NLP, yang dipimpin oleh penggunaan pra-pelatihan mandiri dalam skala besar. Dan kemajuan besar sedang dilakukan di banyak tugas, tidak hanya dalam NLP tetapi juga dalam visi komputer dan bidang AI lainnya. Salah satu ide di balik kemajuan ini adalah bahwa sejumlah besar pengetahuan tentang dunia dapat dipelajari hanya dengan melatih sistem AI ini pada sejumlah besar data yang tidak berlabel. Misalnya, model bahasa dapat mempelajari beberapa pengetahuan umum pengetahuan tentang suatu bahasa dengan cara memprediksi kata-kata yang tersamar secara berulang dalam teks berskala besar. Akibatnya, model bahasa yang telah dilatih sebelumnya ini dapat berfungsi sebagai model dasar, yang dapat dengan mudah diadaptasi untuk menangani tugas-tugas NLP hilir tertentu. Pergeseran paradigma dalam NLP ini telah memungkinkan pengembangan sistem yang sangat kuat untuk pemahaman, pembuatan, dan penalaran bahasa [Manning, 2022]. Namun, penting untuk menyadari bahwa kita masih dalam tahap awal menciptakan sistem yang benar-benar cerdas, dan masih banyak yang harus dilakukan. Meskipun demikian, pelatihan awal skala besar telah dibuka pintu menuju sistem cerdas yang telah lama ingin dikembangkan oleh para peneliti, meskipun beberapa area penelitian utama masih terbuka untuk dieksplorasi, seperti pembelajaran kecerdasan secara efisien dengan menggunakan sumber daya yang cukup. data berukuran kecil dan memperoleh kemampuan penalaran dan perencanaan yang rumit.

Perlu dicatat bahwa bab ini sebagian besar merupakan pengantar dan tidak dapat mencakup semua aspek pra-pelatihan. misalnya, ada banyak metode untuk menyempurnakan model yang telah dilatih sebelumnya, menawarkan cara yang berbeda untuk lebih baik mengadaptasi model ke dalam berbagai situasi. Selain itu, model bahasa yang besar, yang dianggap sebagai salah satu salah satu pencapaian paling signifikan dalam AI dalam beberapa tahun terakhir, dilewati di bagian ini. Kami meninggalkan pembahasan topik-topik tersebut pada bab-bab berikutnya.

BAB 2

Model Generatif

Salah satu kemajuan paling signifikan dalam NLP dalam beberapa tahun terakhir mungkin adalah pengembangan model bahasa (LLM). Hal ini telah membantu menciptakan sistem yang dapat memahami dan menghasilkan bahasa alami seperti manusia. Sistem ini bahkan telah ditemukan mampu bernalar, yang dianggap sebagai masalah AI yang sangat menantang. Dengan pencapaian ini, NLP membuat langkah besar dan memasuki era baru penelitian di mana masalah-masalah sulit sedang dipecahkan, seperti membangun sistem percakapan yang dapat berkomunikasi dengan manusia dengan lancar.

Konsep pemodelan bahasa atau pemodelan bahasa probabilistik berasal dari eksperimen awal yang dilakukan oleh [Shannon \[1951\]](#). Dalam karyanya, model bahasa dirancang untuk memperkirakan predikabilitas bahasa Inggris — seberapa baik huruf berikutnya dari sebuah teks dapat diprediksi ketika huruf N sebelumnya diketahui. Meskipun percobaan Shannon masih bersifat awal, Tujuan dan metode pemodelan bahasa sebagian besar tetap tidak berubah selama beberapa dekade sejak kemudian. Untuk jangka waktu yang cukup lama, terutama sebelum tahun 2010, pendekatan dominan terhadap pemodelan bahasa adalah pendekatan n-gram [\[Jurafsky dan Martin, 2008\]](#). Dalam pemodelan bahasa n-gram, kita memperkirakan probabilitas sebuah kata jika diberikan $n - 1$ kata sebelumnya, dan dengan demikian probabilitas sebuah kata Urutan dapat didekati dengan produk dari serangkaian probabilitas n-gram. Probabilitas ini biasanya diperkirakan dengan mengumpulkan jumlah relatif n-gram yang dihaluskan dalam teks. Sementara Pendekatan seperti ini mudah dan sederhana, dan telah banyak digunakan dalam NLP. Misalnya, keberhasilan pengenalan ucapan statistik modern dan sistem penerjemahan mesin sebagian besar tergantung pada pemanfaatan model bahasa n-gram [\[Jelinek, 1998; Koehn, 2010\]](#).

Penerapan jaringan saraf pada pemodelan bahasa telah lama menarik, tetapi terobosan nyata muncul seiring dengan kemajuan teknik pembelajaran mendalam. Sebuah studi yang banyak dikutip adalah [Bengio et al. \[2003\]](#) bekerja di mana probabilitas n-gram dimodelkan melalui jaringan umpan maju dan dipelajari dengan melatih jaringan secara menyeluruh. Produk sampingan dari model bahasa saraf ini adalah representasi kata-kata yang terdistribusi, yang dikenal sebagai word embeddings. Daripada merepresentasikan kata-kata sebagai variabel diskrit, word embeddings memetakan kata-kata ke dalam variabel bernilai riil berdimensi rendah. vektor, sehingga memungkinkan untuk menghitung makna kata dan n-gram kata dalam ruang representasi yang berkelanjutan. Akibatnya, model bahasa tidak lagi dibebani dengan kutukan dimensionalitas, tetapi dapat mewakili banyak n-gram secara eksponensial melalui jaringan saraf yang kompak dan padat model.

Ide pembelajaran representasi kata melalui model bahasa saraf menginspirasi penelitian selanjutnya penelitian dalam pembelajaran representasi dalam NLP. Namun, pendekatan ini tidak menarik minat yang signifikan dalam mengembangkan sistem NLP dalam beberapa tahun pertama setelah proposalnya. Dimulai sekitar tahun 2012, meskipun, kemajuan telah dibuat dalam mempelajari penyematan kata dari teks skala besar melalui kata sederhana tugas prediksi. Beberapa metode, seperti Word2Vec, diusulkan untuk mempelajarinya secara efektif penanaman, yang kemudian berhasil diterapkan dalam berbagai sistem NLP [\[Mikolov et al., 2013a;b\]](#). Sebagai hasil dari kemajuan ini, para peneliti mulai memikirkan representasi pembelajaran urutan menggunakan model bahasa yang lebih kuat, seperti model berbasis LSTM [\[Sutskever et al., 2014; Peters et al., 2018\]](#). Dan kemajuan dan minat lebih lanjut dalam representasi sekvensi meledak setelah Transformer diusulkan. Bersamaan dengan munculnya Transformer, konsep pemodelan bahasa digeneralisasi untuk mencakup model yang belajar memprediksi kata-kata dengan berbagai cara. Banyak

Model berbasis Transformer yang kuat telah dilatih sebelumnya menggunakan tugas prediksi kata ini, dan berhasil diterapkan ke berbagai tugas hilir [Devlin et al., 2019].

Sesungguhnya, melatih model bahasa pada data berskala besar telah membawa penelitian NLP ke masa-masa yang menarik. Meskipun pemodelan bahasa telah lama dianggap sebagai teknik dasar yang tidak memiliki hubungan langsung dengan tujuan kecerdasan buatan yang diharapkan oleh para peneliti, ini membantu kita melihat munculnya sistem cerdas yang dapat mempelajari tingkat pengetahuan umum tertentu dari prediksi berulang kata-kata dalam teks. Penelitian terbaru menunjukkan bahwa seorang LLM yang terlatih dengan baik dapat menangani sejumlah besar sejumlah tugas dan digeneralisasi untuk melakukan tugas baru dengan upaya adaptasi kecil [Bubeck et al., 2023]. Hal ini menunjukkan langkah menuju bentuk kecerdasan buatan yang lebih maju, dan menginspirasi eksplorasi lebih lanjut untuk mengembangkan model bahasa yang lebih kuat sebagai model dasar.

Dalam bab ini, kami akan membahas konsep dasar LLM generatif. Untuk menyederhanakannya, kami menggunakan istilah model bahasa besar atau LLM untuk merujuk pada model generatif seperti GPT, meskipun istilah ini dapat secara luas mencakup jenis model lain seperti BERT. Kami mulai dengan memberikan pengenalan umum untuk LLM, termasuk langkah-langkah kunci dalam membangun model tersebut. Kami kemudian membahas dua masalah skala LLM: bagaimana LLM dilatih dalam skala besar, dan bagaimana LLM dapat ditingkatkan untuk menangani teks yang sangat panjang. Akhirnya, kami memberikan ringkasan dari diskusi tersebut.

2.1 Pengantar Singkat tentang LLM

Pada bagian ini kami memberikan pengantar tentang ide-ide dasar LLM seperti yang diperlukan untuk sisa bab ini bab dan bab-bab berikutnya. Kita akan menggunakan istilah kata dan token secara bergantian. Keduanya diantaranya mengacu pada unit dasar yang digunakan dalam pemodelan bahasa, meskipun makna aslinya adalah berbeda.

Sebelum menyajikan rinciannya, mari kita pertimbangkan terlebih dahulu cara kerja model bahasa. Tujuan pemodelan bahasa adalah untuk memprediksi probabilitas terjadinya serangkaian token. Misalkan $\{x_0, x_1, \dots, x_m\}$ menjadi urutan token, di mana x_0 adalah simbol awal s (atau SOS) · Kemungkinan dari se- Urutan dapat didefinisikan menggunakan aturan rantai

$$\Pr(x_0, \dots, x_m) = \Pr(x_0) \cdot \Pr(x_1|x_0) \cdot \Pr(x_2|x_0, x_1) \cdots \Pr(x_m|x_0, \dots, x_{m-1}) \quad (2.1)$$

Persamaan (x dan $|$) adalah persamaan yang menyatakan hubungan antara x dan $|$.
saya=0

atau alternatifnya dalam bentuk logaritma

$$\logaritma \Pr(x_0, \dots, x_m) = \sum_{i=0}^m \log \Pr(x_i | x_0, \dots, x_{i-1}) \quad (2.2)$$

Di sini $\Pr(x_i | x_0, \dots, x_{i-1})$ adalah probabilitas token x_i yang diberikan semua token sebelumnya $\{x_0, \dots, x_{i-1}\}$

- 2 · Di era pembelajaran mendalam, pendekatan khas terhadap pemodelan bahasa adalah memperkirakan hal ini

1Simbol awal juga bisa [CLS] mengikuti model BERT.

2Kita asumsikan bahwa ketika $i = 0$, $\Pr(x_i|x_0, \dots, x_{i-1}) = \Pr(x_0) = 1$. Maka $\Pr(x_0, \dots, x_m) = \Pr(x_0) \Pr(x_1, \dots, x_m|x_0) = \Pr(x_1, \dots, x_m|x_0)$.

Konteks	Aturan Prediksi Keputusan	Probabilitas Urutan
aku	B	$\text{argumen } \max_{x2} \Pr(x2/s a)$
sab	C	$\arg \max_{x3} \Pr(x3/s a b) \Pr(s) \cdot \Pr(a/s) \cdot \Pr(b/s a) \cdot \Pr(c/s \text{ sebuah } b)$
sabcd		$\arg \max_{x4} \Pr(x4/s ab c) \Pr(s) \cdot \Pr(a/s) \cdot \Pr(b/s a) \cdot \Pr(c/s \text{ sebuah } b) \cdot \Pr(d/s ab c)$

Tabel 2.1: Ilustrasi pembuatan tiga token *bcd* yang diberi awalan *sa* melalui model bahasa. Pada setiap langkah, model memilih token *xi* dari *V* sehingga $\Pr(xi/x_0, \dots, xi-1)$ dimaksimalkan. Token ini kemudian ditambahkan ke akhir dari urutan konteks. Pada langkah berikutnya, kita ulangi proses yang sama, tetapi berdasarkan konteks baru.

probabilitas menggunakan jaringan saraf dalam. Jaringan saraf yang dilatih untuk menyelesaikan tugas ini menerima urutan token $x_0, \dots, xi-1$ dan menghasilkan distribusi atas kosakata *V* (dilambangkan dengan $\Pr(\cdot|x_0, \dots, xi-1)$). Probabilitas $\Pr(xi|x_0, \dots, xi-1)$ adalah nilai entri ke-i dari $\Pr(\cdot|x_0, \dots, xi-1)$.

Saat menerapkan model bahasa yang terlatih, tugas yang umum adalah menemukan token yang paling mungkin diberikan token konteks sebelumnya. Tugas prediksi token ini dapat dijelaskan sebagai

$$\hat{x}_i = \arg \max_{xi \in V} \Pr(xi|x_0, \dots, xi-1) \quad (2.3)$$

Kita dapat melakukan prediksi kata beberapa kali untuk menghasilkan teks berkelanjutan: setiap kali kita prediksi token terbaik \hat{x}_i , lalu tambahkan token prediksi ini ke konteks untuk memprediksi token berikutnya token \hat{x}_{i+1} . Hal ini menghasilkan proses pembangkitan dari kiri ke kanan yang menerapkan Persamaan (2.1) dan (2.2). mengilustrasikan, perhatikan generasi tiga kata berikut yang diberikan awalan 'a', seperti yang ditunjukkan pada Tabel 2.1. Sekarang kita bahas bagaimana LLM dibangun, dilatih, dan diterapkan.

2.1.1 Transformator Hanya Dekoder

Seperti praktik standar, input dari model bahasa adalah serangkaian token (dilambangkan dengan $\{x_0, \dots, xm\}$). Untuk setiap langkah, token keluaran dibuat, menggeser urutan satu posisi ke depan untuk prediksi berikutnya. Untuk melakukan ini, model bahasa mengeluarkan distribusi $\Pr(\cdot|x_0, \dots, xi-1)$ di setiap posisi *i*, dan token *xi* dipilih menurut distribusi ini.

Model ini dilatih dengan memaksimalkan kemungkinan $\log \prod_{i=1}^M \Pr(xi|x_0, \dots, xi-1)$ ³.

Di sini, kami fokus pada arsitektur Transformer khusus dekoder, karena ini adalah salah satu yang paling populer arsitektur model yang digunakan dalam LLM. Urutan input token diwakili oleh urutan vektor de-dimensi $\{e_0, \dots, em\}$. *ei* adalah jumlah dari token embedding *xi* dan penempatan posisi *i*. Tubuh utama model adalah tumpukan blok Transformer (atau lapisan). Setiap blok Transformer memiliki dua sub-lapisan bertumpuk, satu untuk pemodelan perhatian diri dan satu untuk pemodelan FFN. Sub-lapisan ini dapat didefinisikan menggunakan arsitektur post-norm

$$\text{keluaran} = \text{LNorm}(\text{F}(\text{masukan}) + \text{masukan}) \quad (2.4)$$

³Perhatikan bahwa $\log \Pr(xi|x_0, \dots, xi-1) = \sum_{saya=0}^M \log \Pr(xi|x_0, \dots, xi-1)$ karena $\log \Pr(x_0) = 0$.

atau arsitektur pra-norma

$$\text{keluaran} = \text{LNorm}(F(\text{masukan})) + \text{masukan} \quad (2.5)$$

di mana input dan output menunjukkan input dan output, keduanya berupa matriks $m \times d$. Baris ke- i dari input dan output dapat dilihat sebagai representasi kontekstual dari token ke- i dalam urutan tersebut.

$F(\cdot)$ adalah fungsi inti dari sub-lapisan. Untuk sub-lapisan FFN, $F(\cdot)$ adalah FFN multi-lapisan. Untuk sub-lapisan self-attention, $F(\cdot)$ adalah fungsi self-attention multi-head. Secara umum, self-attention dinyatakan dalam bentuk QKV attention

$$\text{AttQKV}(Q, K, V) = \text{Softmax}(QK^T \frac{\sqrt{d}}{\sqrt{d}} + \text{Masker})V \quad (2.6)$$

di mana Q , K dan V ukuran sedang $\in \mathbb{R}^{m \times d}$ masing-masing adalah kueri, kunci, dan nilai. Penting untuk dicatat bahwa hanya token sebelumnya yang dipertimbangkan saat memprediksi token. Jadi variabel masking Mask $\in \mathbb{R}^{m \times m}$ dimasukkan dalam self-attention untuk mencapai hal ini. Entri (i, k) dari Mask memiliki nilai 0 jika $i \neq k$, dan nilai $\neq \inf$ jika tidak.

Diberikan representasi $H \in \mathbb{R}^{m \times d}$ fungsi perhatian diri multi-kepala dapat didefinisikan sebagai

$$F(H) = \text{Gabungkan}(\text{kepala}1, \dots, \text{kepala}h)W_{\text{kepala}} \quad (2.7)$$

di mana $\text{Merge}(\cdot)$ merepresentasikan penggabungan inputnya, dan $W_{\text{head}} \in \mathbb{R}^{d \times d}$ mewakili sebuah parameter. head_j adalah output dari perhatian QKV pada sub-ruang representasi

$$\text{kepala}_j = \text{AttQKV}(Q[j], K[j], V[j]) \quad (2.8)$$

$Q[j]$, $K[j]$, dan $V[j]$ adalah kueri, kunci, dan nilai yang diproyeksikan ke subruang j melalui linier transformasi

$$Q[j] = HWq \quad (2.9)$$

$$K[j] = HWk \quad (2.10)$$

$$V[j] = HWv \quad (2.11)$$

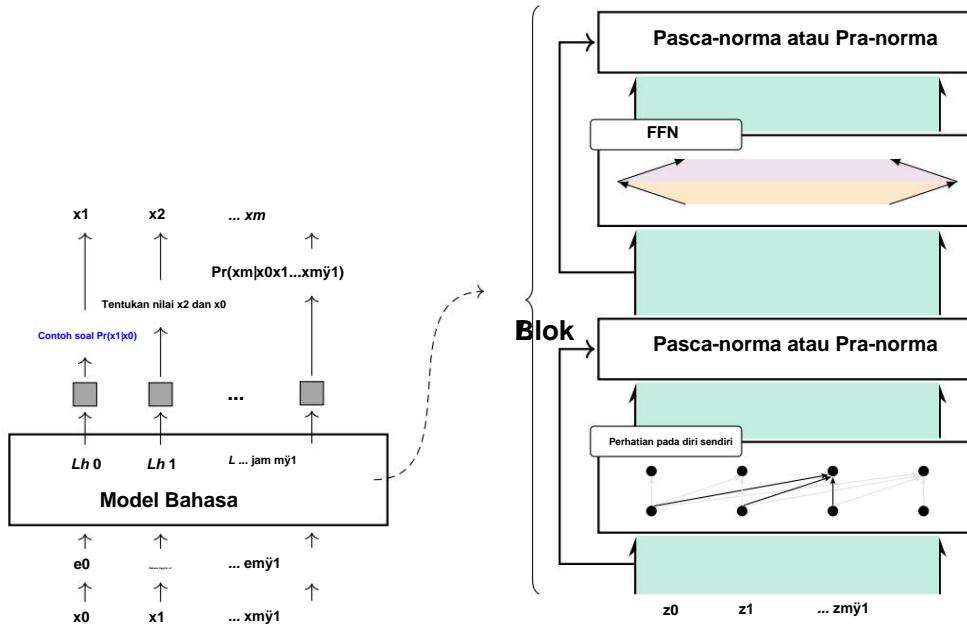
dimana Wq dan $Wv \in \mathbb{R}^{d \times d}$ adalah matriks parameter transformasi.

Misalkan kita memiliki blok L Transformer. Lapisan Softmax dibangun di atas keluaran blok terakhir. Lapisan Softmax mengeluarkan urutan distribusi m atas kosakata, seperti ini

$$\begin{matrix} \Pr(\cdot|x_0, \dots, x_m) \\ \vdots \\ \Pr(\cdot|x_0, x_1) \\ \Pr(\cdot|x_0) \end{matrix} = \text{Softmax}(HLWo) \quad (2.12)$$

di mana HL adalah keluaran blok Transformer terakhir, dan $Wo \in \mathbb{R}^{d \times V}$ adalah matriks parameter.

Gambar 2.1 menunjukkan arsitektur Transformer untuk pemodelan bahasa. Penerapan bahasa ini



Gbr. 2.1: Arsitektur Transformer-decoder untuk pemodelan bahasa. Komponen utamanya adalah blok Transformer yang ditumpuk L , masing-masing terdiri dari sublapisan self-attention dan sublapisan FFN. Untuk mencegah model mengakses konteks yang tepat, variabel masking dimasukkan ke dalam self-attention. Lapisan output menggunakan fungsi Softmax untuk menghasilkan distribusi probabilitas untuk token berikutnya, dengan urutan token sebelumnya. Selama inferensi, model mengambil token yang diprediksi sebelumnya untuk memprediksi token berikutnya, mengulangi proses ini hingga akhir urutan tercapai. $\{z_0, \dots, z_{m-1}\}$ menunjukkan input dari blok Transformer, dan $\{h$ blok Transformer terakhir.

model mengikuti proses autoregresif. Setiap kali model bahasa mengambil token x_i sebagai input dan memprediksi token x_i yang memaksimalkan probabilitas $Pr(x_i | x_0, \dots, x_{i-1})$. Penting untuk dicatat bahwa, meskipun detail implementasinya berbeda, banyak LLM berbagi arsitektur yang sama seperti yang dijelaskan di atas. Model-model ini disebut besar karena kedalamannya dan lebarnya signifikan.

Tabel 2.2 menunjukkan ukuran model untuk beberapa LLM, serta pengaturan modelnya.

2.1.2 Pelatihan LLM

Sekarang anggaplah kita diberi set pelatihan D yang terdiri dari K sekuen. Log-likelihood dari setiap sekuen $x = x_0 \dots x_m$ di D dapat dihitung menggunakan model bahasa.

$$L_j(x) = \log \Pr(x_i | x_0, \dots, x_{i-1}) \quad (2.13)$$

saya=1

Di sini subskrip j yang ditambahkan pada $L(\cdot)$ dan $Pr(\cdot)$ menunjukkan parameter model bahasa. Maka, tujuan pelatihan kemungkinan maksimum didefinisikan sebagai

$$\hat{j} = \arg \max_j L_j(x) \quad (2.14)$$

Pelatihan model bahasa berbasis Transformer dengan tujuan di atas umumnya dipandang sebagai proses pengoptimalan standar untuk jaringan saraf. Hal ini dapat dicapai dengan menggunakan algoritma penurunan gradien, yang didukung secara luas oleh perangkat pembelajaran mendalam yang tersedia di pasaran.

Magister Hukum	# Parameter	Kedalaman L	Lebar d	# Kepala	(Q/KV)
GPT-1 [Radford dkk., 2018]	0,117 miliar	12	768		Tanggal 12/12
GPT-2 [Radford dkk., 2019]	1,5 miliar	48	1.600 orang		25/25
GPT-3 [Brown dkk., 2020]	175B	96	12.288	4.096	tahun 96/96
	7B	32	5.120		32/32
LLaMA2 [Touvron dkk., 2023b]	13B	40	8.192		40/40
	70B	80	4.096		64/64
	8B	32			32/8
LLaMA3/3.1 [Dubey dkk., 2024]	70B	80	8.192		64/8
	405B	126	16.384		128/8
	2B	26	2.304		8/4
Gemma2 [Tim dkk., 2024]	9B	42	3.584		16/8
	37B	46	4.608		tanggal 32/16
	0,5 miliar	24	896		14/2
Qwen2.5 [Yang dkk., 2024]	7B	28	3.584		28/4
	72B	80	8.192		64/8
DeepSeek-V3 [Liu dkk., 2024a]	671B	61	7.168 orang		128/128
	7B	32	4.544		71/71
Elang [Penedo dkk., 2023]	40B	60	8.192		128/128
	180B	80	14.848		232/232
Mistral [Jiang dkk., 2023a]	7B	32	4.096 orang		32/32

Tabel 2.2: Perbandingan beberapa LLM dalam hal ukuran model, kedalaman model, lebar model, dan jumlah kepala (a/b berarti kepala a untuk kueri dan kepala b untuk kunci dan nilai).

yang mengejutkan, hasil yang lebih baik terus diperoleh seiring dengan berkembangnya model bahasa menjadi lebih model intensif komputasi dan dilatih pada kumpulan data yang lebih besar [Kaplan et al., 2020]. Keberhasilan ini telah mendorong para peneliti NLP untuk terus meningkatkan data pelatihan dan ukuran model dalam untuk membangun model bahasa yang lebih kuat.

Namun, seiring dengan semakin besarnya model bahasa, kita menghadapi tantangan pelatihan baru, yaitu secara signifikan mengubah masalah dibandingkan dengan melatih model yang relatif kecil. Salah satu dari ini Tantangan muncul dari kebutuhan akan sistem terdistribusi skala besar untuk mengelola data, model parameter, rutinitas pelatihan, dan sebagainya. Mengembangkan dan memelihara sistem tersebut memerlukan sejumlah besar pekerjaan di bidang rekayasa perangkat lunak dan perangkat keras, serta keahlian dalam Masalah terkait adalah ketika pelatihan ditingkatkan, kita memerlukan lebih banyak sumber daya komputasi untuk memastikan proses pelatihan dapat diselesaikan dalam waktu yang dapat diterima. Misalnya, umumnya membutuhkan ratusan atau ribuan GPU untuk melatih LLM dengan puluhan miliar parameter dari awal. Persyaratan ini secara drastis meningkatkan biaya pelatihan model tersebut, terutama mengingat banyaknya pelatihan yang diperlukan saat model ini dikembangkan. Selain itu, dari perspektif pembelajaran mendalam, proses pelatihan dapat menjadi tidak stabil jika jaringan saraf sangat dalam dan/atau ukurannya sangat besar. Sebagai tanggapan, kami biasanya perlu memodifikasi model tersebut arsitektur untuk mengadaptasi LLM ke pelatihan skala besar. Di Bagian 2.2 kami akan menyajikan lebih banyak diskusi pada masalah ini.

2.1.3 Penyempurnaan LLM

Setelah kita melatih LLM terlebih dahulu, kita dapat menerapkannya untuk melakukan berbagai tugas NLP. Secara tradisional, model bahasa digunakan sebagai komponen sistem lain, misalnya, model tersebut diterapkan secara luas untuk menilai terjemahan dalam sistem terjemahan mesin statistik. Sebaliknya, dalam AI generatif, LLM dianggap sebagai sistem yang lengkap dan digunakan untuk mengatasi masalah NLP dengan memanfaatkan sifat pembangkitannya. Pendekatan yang umum adalah mendeskripsikan tugas yang ingin kita tangani dalam teks dan kemudian meminta LLM untuk menghasilkan teks berdasarkan deskripsi ini. Ini adalah tugas pembangkitan teks standar di mana kita melanjutkan atau menyelesaikan teks mulai dari konteks yang diberikan.

Secara lebih formal, misalkan $x = x_0 \dots x_m$ menunjukkan urutan token konteks yang diberikan oleh pengguna, dan $y = y_1 \dots y_n$ menunjukkan urutan token yang mengikuti konteks. Dengan demikian, inferensi LLM dapat didefinisikan sebagai masalah menemukan urutan y yang paling mungkin berdasarkan x :

$$\begin{aligned}
 &= \arg \max_{\text{sayang}} \log \Pr(y|x) \\
 &\quad \text{Kamus} \\
 &\quad \quad N \\
 &= \arg \max_{\text{sayang}} y \quad \log \Pr(y_i | x_0, \dots, x_m, y_1, \dots, y_{i-1}) \\
 &\quad \quad \text{sayang}=1
 \end{aligned} \tag{2.15}$$

Di Sini $\log \Pr(y_i | x_0, \dots, x_m, y_1, \dots, y_{i-1})$ pada dasarnya menyatakan hal yang sama dengan sisi kanan Persamaan (2.2). Persamaan ini memodelkan probabilitas logaritma untuk memprediksi token dari posisi $m + 1$, bukan dari posisi 0. Sepanjang bab ini dan bab-bab berikutnya, kita akan menggunakan variabel terpisah x dan y untuk membedakan input dan output dari LLM, meskipun keduanya dapat dilihat sebagai sub-urutan dari urutan yang sama. Dengan mengadopsi notasi tersebut, kita melihat bahwa bentuk persamaan di atas sangat mirip dengan yang digunakan dalam model pembuatan teks lain dalam NLP, seperti model penerjemahan mesin saraf.

Untuk mengilustrasikan bagaimana LLM diterapkan, pertimbangkan masalah penentuan grammatikalitas untuk kalimat tertentu. Kita dapat mendefinisikan templat seperti ini

{*kalimat*}

Pertanyaan: Apakah kalimat ini benar secara tata bahasa?

Menjawab: _____

Berikut ini merupakan teks yang ingin kita buat. {*sentence*} adalah variabel pengganti yang akan diganti dengan kalimat sebenarnya yang diberikan oleh pengguna. Misalnya, anggaplah kita memiliki kalimat "John tampak bahagia hari ini.". Kita dapat mengganti {*sentence*} dalam templat dengan kalimat ini untuk mendapatkan masukan ke model bahasa.

John nampak gembira hari ini.

Pertanyaan: Apakah kalimat ini benar secara tata bahasa?

Menjawab: _____

Untuk melakukan tugas tersebut, model bahasa diberikan konteks $x =$ "John tampaknya bahagia hari ini.\n" Pertanyaan : Apakah kalimat ini secara tata bahasa benar? Jawaban :"4 . Kemudian menghasilkan hal berikut

⁴\n adalah karakter khusus yang digunakan untuk jeda baris.

teks sebagai jawaban, berdasarkan konteksnya. Misalnya, model bahasa mungkin mengeluarkan “Ya” (yaitu, y = “Ya”) jika teks ini adalah teks yang memiliki probabilitas prediksi maksimum dalam konteks ini.

Demikian pula, kita dapat menentukan lebih banyak template untuk menangani tugas-tugas lainnya. Misalnya, kita dapat menerjemahkan kalimat bahasa Inggris ke bahasa Mandarin menggunakan template berikut

{*kalimat*}

Pertanyaan: Apa terjemahan bahasa Mandarin dari kalimat bahasa Inggris ini?

Menjawab: _____

atau menggunakan template seperti instruksi

{*kalimat*}

Terjemahkan kalimat ini dari bahasa Inggris ke bahasa Mandarin.

atau menggunakan templat seperti kode.

[src-lang] = Bahasa Inggris [tgt-lang] = Bahasa Mandarin [input] = {*kalimat*}

[keluaran] = _____

Template di atas menyediakan metode yang sederhana namun efektif untuk “meminta” satu LLM untuk melakukan berbagai tugas tanpa mengadaptasi struktur model. Namun, pendekatan ini memerlukan bahwa LLM dapat mengenali dan mengikuti instruksi atau pertanyaan. Salah satu cara untuk melakukan ini adalah dengan memasukkan contoh pelatihan dengan instruksi dan respons yang sesuai ke dalam pra-pelatihan. dataset. Meskipun metode ini mudah, membangun dan melatih LLM dari awal membutuhkan komputasi yang mahal. Selain itu, membuat data yang mengikuti instruksi efektif untuk pra-pelatihan membutuhkan sejumlah besar data tersebut, namun pengumpulan data berlabel dalam skala besar untuk semua tugas bunga sangatlah sulit.

Metode kedua, yang telah menjadi standar de facto dalam penelitian baru-baru ini, adalah mengadaptasi LLM melalui fine-tuning. Dengan demikian, kemampuan prediksi token yang dipelajari dalam fase pra-pelatihan dapat digeneralisasikan untuk menyelesaikan tugas-tugas baru. Ide di balik fine-tuning adalah bahwa beberapa pengetahuan umum pengetahuan bahasa telah diperoleh dalam pra-pelatihan, tetapi kita memerlukan mekanisme untuk mengaktifkan pengetahuan ini untuk menerapkannya pada tugas-tugas baru. Untuk mencapai hal ini, kita dapat sedikit menyempurnakan parameter model menggunakan data yang mengikuti instruksi. Pendekatan ini disebut penyempurnaan instruksi.

Contoh fine-tuning instruksi, yang direpresentasikan oleh serangkaian token, dapat dilihat sebagai tuple yang terdiri dari input dan output yang diinginkan. Di sini, input mencakup instruksi, informasi sistem (atau awalan sistem), dan informasi lain yang diberikan pengguna⁵ perhatikan contoh berikut (teks biru = masukan dan teks yang digarisbawahi = keluaran).

Untuk mengilustrasikan,

⁵ Informasi sistem mengacu pada serangkaian token yang ditambahkan di awal input untuk memandu perilaku LLM, misalnya, Anda adalah asisten yang membantu dan tidak boleh mengeluarkan konten yang beracun.

Bacalah kalimat berikut, dan jawablah apakah tata bahasanya benar.

LLM adalah model yang kuat tetapi mahal untuk dibangun.

Ya

LLM adalah model yang kuat tetapi mahal untuk dibangun.

Apakah kalimat ini masuk akal secara tata bahasa?

Pilihan Jawaban

- Ya

- TIDAK

Ya

Ya atau Tidak? Ya = kalimat berikut ini benar secara tata bahasa. Tidak = mengandung kesalahan tata bahasa.

LLM adalah model yang kuat tetapi mahal untuk dibangun.

Ya

Semua sampel ini menggambarkan tugas klasifikasi biner yang sama, tetapi dengan instruksi yang berbeda.

Untuk meningkatkan keragaman dalam penyempurnaan data dan membuat LLM dapat digeneralisasi, kita dapat mendefinisikan lebih banyak tugas menggunakan instruksi.

Tulis cerita pendek tentang tiga karakter.

Karakter 1: Seorang anak laki-laki bernama Andy.

Karakter 2: Anjing Andy.

Karakter 3: Teman Andy, Jane.

Di jantung kota yang ramai, Andy, anjingnya yang setia, Rocket, dan temannya, Jane,

menemukan taman tersembunyi di bawah cahaya senja. Dikelilingi oleh pepohonan tua dan

desiran angin, mereka berbagi mimpi dan rahasia, menjalin ikatan yang tak terpisahkan. Saat

malam semakin larut, mereka membuat perjanjian di bawah bintang-bintang untuk selalu

menemukan keajaiban dalam hal-hal yang biasa saja. Bersama-sama, mereka menyadari

bahwa petualangan sejati terletak pada saat-saat yang dihabiskan bersama teman-teman, dan

setiap hari adalah harta karun yang menunggu untuk ditemukan.

Tunjukkan saya resep membuat es krim.

Bahan-bahan:

2 cangkir krim kental; 1 cangkir susu; 3/4 cangkir gula; 2 sdt ekstrak vanila;

Sedikit garam.

Tangga:

1. Kocok semua bahan sampai gula larut.
2. Dinginkan campuran selama 2 jam dalam lemari es.
3. Bekukan dalam piring dangkal, aduk setiap 30 menit, 3-4 kali.
4. Bekukan padat selama 2 jam.
5. Sajikan es krim buatan Anda.

Jika Anda membeli 5 buah apel dan setiap apel harganya \$1,20, berapa jumlah total yang Anda belanjakan?

Rp 6,00

Tulis program Python untuk menghitung jumlah kuadrat angka-angka berikut.

1 2 10 -9 78

angka = [1,2,10,-9,78]

jumlah_kuadrat = jumlah(x**2 untuk x dalam angka)

cetak(jumlah_kuadrat)

Untuk memperoleh kemampuan mengikuti instruksi, diperlukan sejumlah data fine-tuning. Ini data mungkin mencakup beragam instruksi dan kemungkinan respons. Telah ditemukan bahwa penskalaan sejumlah tugas fine-tuning bermanfaat untuk meningkatkan kinerja LLM [Chung et al., 2022]. Perlu dicatat bahwa meskipun penyempurnaan data lebih lanjut lebih baik, jumlah data ini umumnya besar yang lebih kecil dari data pra-pelatihan. Misalnya, LLM dapat disetel dengan baik dengan puluhan atau ratusan ribu sampel, atau bahkan lebih sedikit jika sampel ini memiliki nilai tinggi kualitas [Zhou et al., 2023a; Chen et al., 2023b], sedangkan pra-pelatihan model tersebut mungkin memerlukan miliaran atau triliunan token, yang mengakibatkan permintaan komputasi yang jauh lebih besar dan waktu yang lebih lama waktu pelatihan [Touvron et al., 2023a].

Perlu dicatat juga bahwa kita tidak boleh mengharapkan data fine-tuning mencakup semua tugas hilir yang ingin kita terapkan LLM. Pemahaman umum tentang bagaimana pra-pelatihan + pendekatan fine-tuning bekerja dengan baik karena LLM telah memperoleh pengetahuan untuk memahami instruksi dan menghasilkan respons pada fase pra-pelatihan. Namun, kemampuan ini belum sepenuhnya diaktifkan sampai kita memperkenalkan beberapa bentuk pengawasan. Perilaku mengikuti instruksi secara umum muncul saat kita menyempurnakan model dengan jumlah data berlabel yang relatif kecil. Hasilnya, kita dapat mencapai tingkat pembelajaran zero-shot: model yang disetel dengan baik dapat menangani tugas baru yang mereka belum dilatih secara eksplisit atau disempurnakan untuk [Sanh et al., 2022; Wei et al., 2022a]. Kemampuan pembelajaran zero-shot ini membedakan LLM generatif dari model pra-terlatih sebelumnya seperti BERT, yang terutama disetel untuk tugas-tugas tertentu.

Setelah kami menyiapkan kumpulan data yang dijelaskan berdasarkan instruksi, proses penyempurnaan adalah relatif sederhana. Proses ini dapat dilihat sebagai proses pelatihan standar sebagai pra-pelatihan, tetapi pada dataset pelatihan yang jauh lebih kecil. Biarkan Dtune menjadi dataset fine-tuning dan γ menjadi parameter model

dioptimalkan melalui pra-pelatihan. Kita dapat memodifikasi Persamaan (2.14) untuk mendapatkan tujuan fine-tuning

$$\tilde{y} = \arg \max_{\text{sample} \in \mathcal{Y}} L(\tilde{y} + (\text{sample})) \quad (2.16)$$

Di sini \tilde{y} menunjukkan parameter optimal. Penggunaan notasi $\tilde{y} +$ berarti bahwa fine-tuning dimulai dengan parameter yang telah dilatih sebelumnya \tilde{y} .

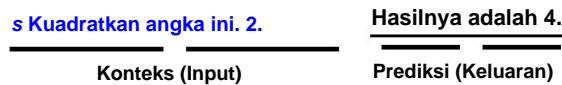
Untuk setiap sampel $y \in \mathcal{Y}$, kita membaginya menjadi segmen input x_{sample} dan segmen output y_{sample} , yaitu,

$$\text{sample} = [y_{\text{sample}}, x_{\text{sample}}] \quad (2.17)$$

Kemudian kita mendefinisikan fungsi kerugian menjadi

$$L(\tilde{y} + (\text{sample})) = \tilde{y} \log P(\tilde{y} + (\text{sample}) | \text{sample}, x) \quad (2.18)$$

Dengan kata lain, kami menghitung kerugian pada sub-urutan y_{sample} , dan bukan pada keseluruhan urutan. Dalam implementasi praktis back-propagation untuk persamaan ini, urutan $[y_{\text{sample}}, x_{\text{sample}}]$ dibangun dalam forward pass seperti biasa. Namun, dalam backward pass, gradien kesalahan disebarluaskan kembali hanya melalui bagian jaringan yang sesuai dengan y_{sample} , membiarkan sisa jaringan tidak berubah. Sebagai contoh, perhatikan urutan



Kerugian dihitung dan disebarluaskan kembali hanya untuk Hasilnya adalah 4 ..

Penyetelan instruksi juga memerlukan pekerjaan rekayasa yang substansial. Untuk mencapai hasil yang memuaskan, seseorang dapat bereksperimen dengan pengaturan yang berbeda dari laju pembelajaran, ukuran batch, jumlah langkah penyetelan, dan sebagainya. Ini biasanya memerlukan banyak pengujian dan evaluasi penyetelan. Biaya dan upaya eksperimental penyetelan tetap penting dan tidak boleh diabaikan, meskipun jauh lebih rendah daripada fase pra-pelatihan.

Sementara kami fokus pada penyempurnaan instruksi sebagai contoh ilustrasi di sini, teknik penyempurnaan memainkan peran penting dalam mengembangkan berbagai LLM dan lebih banyak digunakan. Contohnya termasuk penyempurnaan LLM sebagai chatbot yang menggunakan data dialog, dan mengadaptasi model ini untuk menangani urutan yang sangat panjang. Penerapan penyempurnaan yang luas telah mendorong para peneliti untuk meningkatkan teknik ini, seperti merancang algoritma penyempurnaan yang lebih efisien. Sementara penelitian tentang penyempurnaan membawa hasil, di bagian ini kami hanya memberikan gambaran tentang langkah-langkah utama yang terlibat. Kami akan melihat diskusi yang lebih terperinci tentang topik ini di bab-bab berikut.

2.1.4 Menyelaraskan LLM dengan Dunia

Penyempurnaan instruksi menyediakan cara sederhana untuk mengadaptasi LLM ke tugas-tugas yang dapat didefinisikan dengan baik. Masalah ini secara luas dapat dikategorikan sebagai masalah penyelarasan. Di sini, penyelarasan disebut sebagai proses membimbing LLM untuk berperilaku dengan cara yang selaras dengan niat manusia. Bimbingan dapat berasal dari data berlabel, umpan balik manusia, atau bentuk preferensi manusia lainnya. Misalnya,

Kami ingin LLM tidak hanya akurat dalam mengikuti instruksi, tetapi juga tidak memihak, jujur, dan tidak berbahaya. Jadi, kami perlu mengawasi model agar sesuai dengan nilai dan harapan manusia. Contoh umum adalah ketika kami bertanya kepada LLM tentang cara membuat senjata, mereka mungkin memberikan daftar langkah-langkah utama untuk melakukannya jika tidak diselaraskan dengan hati-hati. Namun, model yang bertanggung jawab harus mengenali dan menghindari menanggapi permintaan informasi yang berbahaya atau ilegal. Penyelarasan dalam kasus ini sangat penting untuk memastikan bahwa LLM bertindak secara bertanggung jawab dan sesuai dengan.

Konsep yang terkait dengan penyelarasan adalah keamanan AI. Salah satu tujuan utama AI adalah membangun sistem cerdas yang aman dan bermanfaat secara sosial. Untuk mencapai tujuan ini, kita harus menjaga sistem ini tetap tangguh, aman, dan subjektif, dalam kondisi penggunaan di dunia nyata, bahkan dalam kondisi penyalahgunaan atau penggunaan yang merugikan. Untuk LLM, keamanan dapat ditingkatkan dengan menyelaraskannya dengan panduan manusia yang tepat, seperti data berlabel manusia dan interaksi dengan pengguna selama aplikasi.

Penyelarasan sulit dilakukan karena nilai dan harapan manusia beragam dan berubah-ubah. Terkadang, sulit untuk menjelaskan secara tepat apa yang diinginkan manusia, kecuali kita melihat respons LLM terhadap permintaan pengguna. Hal ini membuat penyelarasan tidak lagi menjadi masalah penyetelan LLM pada tugas yang telah ditentukan sebelumnya, tetapi masalah yang lebih besar dalam melatihnya dengan interaksi dengan dunia nyata.

Akibat kekhawatiran dalam mengendalikan sistem AI, telah terjadi lonjakan penelitian tentang masalah penyelarasan untuk LLM. Biasanya, dua langkah penyelarasan diadopsi setelah LLM dilatih terlebih dahulu pada data tak berlabel berskala besar.

- **Penyetelan Halus Terbimbing (SFT).** Ini melibatkan pelatihan lanjutan LLM yang telah dilatih sebelumnya pada data baru yang berorientasi pada tugas dan diberi label. Teknik SFT yang umum digunakan adalah penyetelan halus instruksi. Seperti yang dijelaskan dalam subbagian sebelumnya, dengan belajar dari data yang diberi anotasi respons instruksi, LLM dapat menyesuaikan diri dengan perilaku yang dimaksudkan untuk mengikuti instruksi, sehingga menjadi mampu melakukan berbagai tugas yang dijelaskan instruksi. Penyetelan halus terbimbing dapat dilihat sebagai mengikuti paradigma pra-pelatihan + penyetelan halus, dan menawarkan metode yang relatif mudah untuk mengadaptasi LLM.
- **Belajar dari Umpan Balik Manusia.** Setelah LLM menyelesaikan pra-pelatihan dan penyempurnaan yang diawasi, LLM dapat digunakan untuk menanggapi permintaan pengguna jika diminta dengan tepat. Namun, model ini dapat menghasilkan konten yang tidak berdasarkan fakta, bias, atau berbahaya. Untuk membuat LLM lebih selaras dengan pengguna, satu pendekatan sederhana adalah belajar langsung dari umpan balik manusia. Misalnya, berdasarkan beberapa instruksi dan masukan yang diberikan oleh pengguna, para ahli diminta untuk mengevaluasi seberapa baik model merespons sesuai dengan preferensi dan minat mereka. Umpan balik ini kemudian digunakan untuk melatih LLM lebih lanjut agar lebih selaras.

Metode umum untuk belajar dari umpan balik manusia adalah dengan menganggapnya sebagai masalah pembelajaran penguatan (RL), yang dikenal sebagai pembelajaran penguatan dari umpan balik manusia (RLHF) [Ouyang et al., 2022]. Metode RLHF awalnya diusulkan untuk mengatasi masalah pengambilan keputusan berurutan umum [Christiano et al., 2017], dan kemudian berhasil digunakan dalam pengembangan model seri GPT [Stiennon et al., 2020]. Sebagai pendekatan pembelajaran penguatan, tujuan RLHF adalah mempelajari kebijakan dengan memaksimalkan beberapa imbalan dari lingkungan. Secara khusus, dua komponen dibangun dalam RLHF:

- **Agen.** Agen, juga disebut agen LM, adalah LLM yang ingin kita latih. Agen ini beroperasi dengan berinteraksi dengan lingkungannya: ia menerima teks dari lingkungan dan

menghasilkan teks lain yang dikirim kembali ke lingkungan. Kebijakan agen adalah fungsi yang ditentukan oleh LLM, yaitu $\text{Pr}(y|x)$.

- **Model Penghargaan.** Model penghargaan merupakan proksi dari lingkungan. Setiap kali agen menghasilkan urutan keluaran, model penghargaan memberikan urutan keluaran ini skor numerik (yaitu, penghargaan). Skor ini memberi tahu agen seberapa baik urutan keluaran tersebut.

Dalam RLHF, kita perlu melakukan dua tugas pembelajaran: 1) pembelajaran model penghargaan, yang melibatkan pelatihan model penghargaan menggunakan umpan balik manusia pada output agen, dan 2) pembelajaran kebijakan, yang melibatkan pengoptimalan kebijakan yang dipandu oleh model penghargaan menggunakan algoritma pembelajaran penguatan. Berikut ini adalah uraian singkat dari langkah-langkah utama yang terlibat dalam RLHF.

- Menyusun kebijakan awal menggunakan pelatihan awal dan penyempurnaan instruksi.
- Gunakan kebijakan untuk menghasilkan beberapa keluaran untuk setiap masukan, lalu kumpulkan umpan balik manusia pada keluaran ini (misalnya perbandingan keluaran).
- Pelajari model penghargaan dari umpan balik manusia.
- Menyempurnakan kebijakan dengan pengawasan dari model penghargaan.

Gambar 2.2 menunjukkan gambaran umum RLHF. Mengingat bagian ini hanya berfungsi sebagai pengantar singkat tentang konsep LLM, pembahasan terperinci tentang teknik RLHF tidak akan disertakan. Sebagai gantinya, kami mengilustrasikan ide dasar di balik RLHF menggunakan contoh sederhana.

Misalkan kita telah melatih LLM melalui pra-pelatihan dan penyempurnaan instruksi. LLM ini digunakan untuk menanggapi permintaan dari pengguna. Misalnya, pengguna dapat memasukkan

Bagaimana saya bisa menjalani kehidupan yang lebih ramah lingkungan?

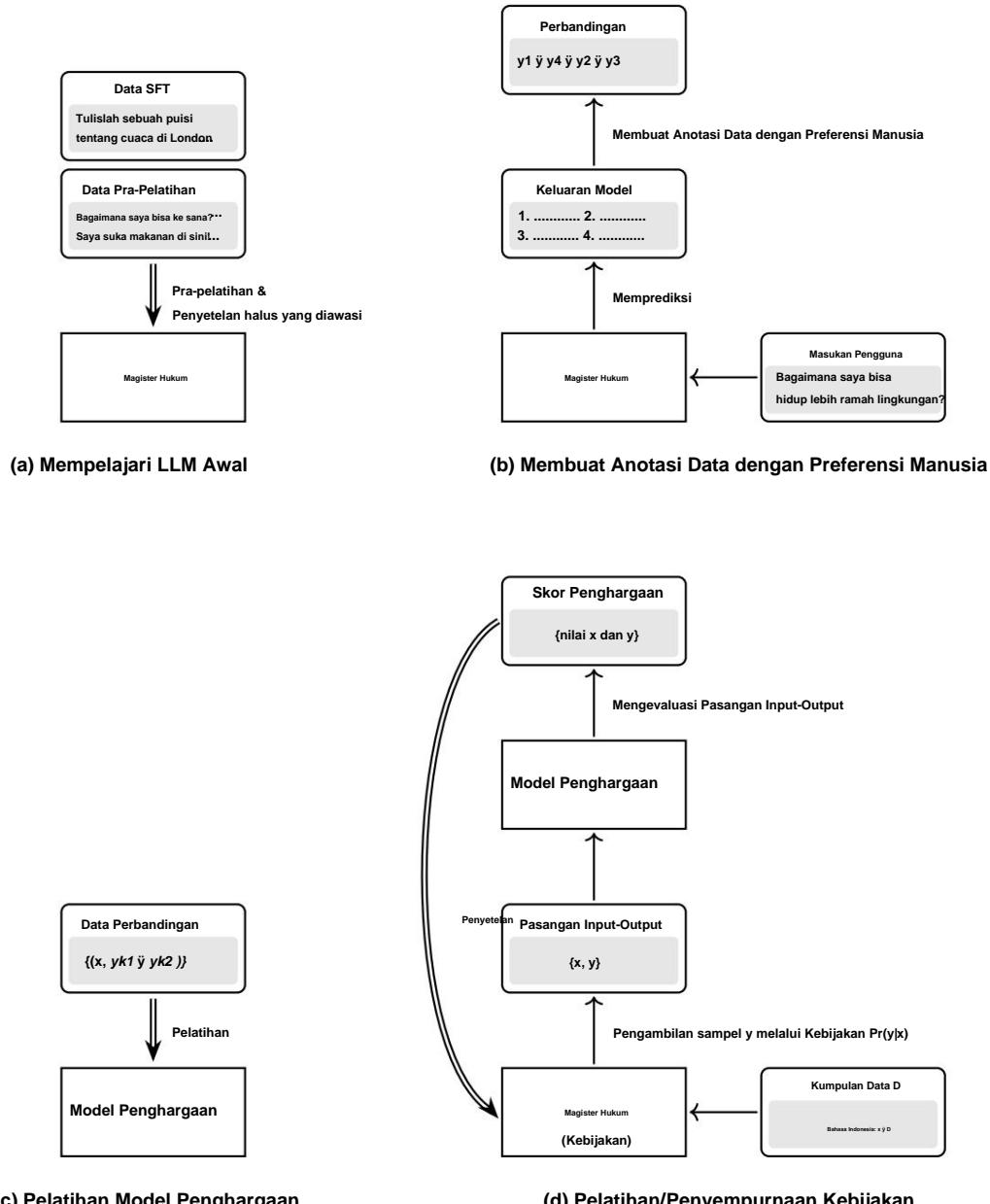
Kami menggunakan LLM untuk menghasilkan 4 keluaran yang berbeda (dilambangkan dengan $\{y_1, \dots, y_4\}$) dengan mengambil sampel ruang keluaran

Output 1 (y_1): Pertimbangkan untuk beralih ke kendaraan listrik atau sepeda sebagai penganti mobil tradisional untuk mengurangi emisi karbon dan melindungi planet kita.

Output 2 (y_2): Mengadopsi gaya hidup minimalis. Memiliki lebih sedikit barang untuk mengurangi konsumsi dan dampak lingkungan dari produksi dan pembuangan.

Output 3 (y_3): Keluar dari jaringan. Hasilkan energi terbarukan Anda sendiri dan kumpulkan air hujan menjadi sepenuhnya mandiri dan mengurangi ketergantungan pada sumber daya yang tidak terbarukan.

Output 4 (y_4): Mendukung produk pertanian lokal untuk mengurangi jejak karbon mengangkut makanan, sambil menikmati makanan segar dan sehat.



Gbr. 2.2: Tinjauan umum RLHF. Ada 4 langkah utama yang terlibat: a) melatih LLM awal (yaitu, kebijakan) menggunakan pra-pelatihan dan penyempurnaan terbimbing; b) mengumpulkan data preferensi manusia dengan memberi peringkat pada keluaran LLM; c) melatih model penghargaan menggunakan hasil pemeringkatan; d) penyempurnaan RL kebijakan berdasarkan model penghargaan. Panah garis ganda berarti pelatihan atau penyempurnaan.

Kami kemudian meminta para anotator untuk mengevaluasi keluaran ini. Salah satu cara yang mudah adalah dengan menetapkan skor penilaian untuk setiap keluaran. Dalam kasus ini, masalah pembelajaran model penghargaan dapat dibingkai sebagai tugas melatih model regresi. Namun, memberikan skor numerik pada keluaran LLM bukanlah tugas yang mudah bagi para anotator. Biasanya sulit untuk merancang standar anotasi yang dapat disetujui dan diikuti dengan mudah oleh semua anotator. Metode alternatif, yang lebih populer dalam pengembangan LLM, adalah memberi peringkat pada keluaran ini. Misalnya, peringkat yang mungkin untuk keluaran di atas adalah

Model penghargaan kemudian dilatih menggunakan hasil pemeringkatan ini. Secara umum, model penghargaan dalam RLHF adalah model bahasa yang memiliki arsitektur yang sama dengan LLM target, tetapi dengan ukuran model yang lebih kecil. Dengan masukan x dan keluaran yk , kami menggabungkan keduanya untuk membentuk urutan $seq_k = [x, yk]$. Urutan ini diproses dari kiri ke kanan menggunakan decoding paksa. Karena setiap posisi hanya dapat mengakses konteks kirinya dalam pemodelan bahasa, output dari lapisan Transformer paling atas pada posisi pertama tidak dapat digunakan sebagai representasi urutan tersebut. Sebaliknya, simbol khusus (misalnya, ls) ditambahkan ke akhir urutan, dan output yang sesuai dari tumpukan lapisan Transformer dianggap sebagai representasi dari seluruh urutan. Lapisan output, seperti lapisan transformasi linier, dibangun di atas representasi ini untuk menghasilkan hadiah, dilambangkan dengan $R(seq_k)$ atau $R(x, yk)$.

Kami melatih model penghargaan ini menggunakan kerugian peringkat. Misalnya, fungsi kerugian peringkat berpasangan dapat ditulis dalam bentuk

$$\text{Kerugian } \hat{y}(Dr) = \sum E(x, yk_1, yk_2) \cdot \log(\text{Sigmoid}(R(\hat{y}(x, yk_1) \hat{y} R(y, yk_2)))) \quad (2.19)$$

di mana \hat{y} mewakili parameter model imbalan, dan Dr mewakili sekumpulan tupel masukan dan sepasang keluaran. $(x, yk_1, yk_2) \in Dr$ adalah operasi pengambilan sampel yang mengambil sampel (x, yk_1, yk_2) dari Dr dengan beberapa probabilitas. Sebagai contoh, anggaplah kita pertama-tama mengambil masukan model x dengan distribusi seragam dan kemudian mengambil sepasang keluaran model dengan probabilitas $yk_1 \neq yk_2$ jika diberikan x (dilambangkan dengan $\Pr(yk_1 \neq yk_2 | x)$). Fungsi kerugian yang sesuai diberikan oleh

$$\begin{aligned} & \text{Kerugian } \hat{y}(Dr) \\ &= \sum \Pr(x) \cdot \Pr(yk_1 \neq yk_2 | x) \cdot \log(\text{Sigmoid}(R(\hat{y}(x, yk_1) \hat{y} R(y, yk_2)))) \\ &\quad - \frac{1}{\Pr(yk_1 \neq yk_2 | x)} \Pr(yk_1 \neq yk_2 | x) \cdot \log(\text{Sigmoid}(R(\hat{y}(x, yk_1) \hat{y} R(y, yk_2)))) \end{aligned} \quad (2.20)$$

di mana K mewakili jumlah masukan model yang terlibat dalam pengambilan sampel. Meskipun bentuk fungsi ini mungkin tampak rumit, idenya sederhana: kita memberi penalti pada model jika peringkat yang diprediksi dari dua keluaran berbeda dari peringkat yang diberi label manusia. Sebaliknya, model menerima bonus, jika peringkat yang diprediksi cocok dengan peringkat yang diberi label manusia.

Kita dapat melatih model penghargaan dengan meminimalkan kerugian peringkat di atas

$$\hat{y} = \arg \min_{\hat{y}} \text{Kerugian } \hat{y}(Dr) \quad (2.21)$$

Model yang dihasilkan $R(\hat{y})$ dapat digunakan untuk mengevaluasi setiap pasangan masukan dan keluaran yang diberikan. Perhatikan bahwa meskipun model penghargaan dilatih menggunakan tujuan berbasis peringkat, model ini digunakan untuk penilaian. Hal ini memungkinkannya memberikan sinyal pengawasan berkelanjutan, yang sangat bermanfaat untuk melatih model lainnya.

Sekarang kita beralih ke masalah pembelajaran kebijakan. Sasaran yang umum diadopsi adalah memaksimalkan imbalan pada serangkaian pasangan masukan-keluaran. Mengikuti bentuk analog dari Persamaan (2.16), kita memperoleh sasaran sederhana untuk penyempurnaan RL

$$\hat{y} = \arg \max_{\hat{y}} E(x, y\hat{y} +) \cdot \Pr(\hat{y}(x, y\hat{y} +)) \quad (2.22)$$

dimana parameter optimal \hat{y} diperoleh dengan melakukan fine-tuning terhadap parameter yang telah dilatih sebelumnya \hat{y} . $\Pr(\hat{y}(x, y\hat{y} +))$

dataset fine-tuning RL. Untuk setiap sampel (x, y^*) , x diambil sampelnya dari dataset yang telah disiapkan urutan masukan, dan y^* diambil sampelnya dari distribusi $\Pr^*(y|x)$ yang diberikan oleh kebijakan.

Dalam praktiknya, algoritma pembelajaran penguatan yang lebih maju, seperti pengoptimalan kebijakan proksimal (PPO), sering digunakan untuk mencapai pelatihan yang lebih stabil, serta kinerja yang lebih baik.

Kami serahkan pembahasan detail algoritma pembelajaran penguatan pada bagian berikut buku ini di mana RLHF digunakan secara luas untuk penyelesaian.

Pertanyaan menarik muncul di sini: mengapa tidak mempertimbangkan pembelajaran dari preferensi manusia sebagai masalah pembelajaran terbimbing standar? Pertanyaan ini terkait erat dengan pertanyaan yang kami sebutkan sebelumnya diskusi tentang kesulitan anotasi data. Seringkali, mendeskripsikan nilai dan tujuan manusia merupakan tantangan, dan bahkan lebih sulit bagi manusia untuk memberikan keluaran yang selaras. Sebagai alternatifnya, membuat anotasi preferensi dari daftar keluaran model yang diberikan menawarkan tugas yang lebih sederhana. dengan melakukan hal ini, kita dapat membuat model yang memahami preferensi manusia, yang kemudian dapat digunakan sebagai model penghargaan untuk kebijakan pelatihan. Dari perspektif pembelajaran mesin, RLHF sangat berguna untuk skenario di mana perilaku agen yang diinginkan sulit untuk ditunjukkan tetapi dapat dengan mudah dikenali oleh manusia. Keuntungan lain dari RLHF adalah kemampuannya untuk mengeksplorasi ruang sampel. Dengan menggunakan teknik pengambilan sampel, model yang dilatih dengan pembelajaran penguatan dapat menjelajah melampaui kumpulan data beranotasi untuk menjelajahi sampel tambahan. Kemampuan eksplorasi ini memungkinkan RLHF untuk menemukan kebijakan yang berpotensi menguntungkan yang tidak langsung terlihat dari data yang diberi label saja.

2.1.5 Meminta LLM

Sejauh ini kami telah menunjukkan bahwa LLM dapat digunakan untuk melakukan berbagai tugas dengan memberikan mereka petunjuk. Tidak ada batasan pada petunjuk ini, yang dapat mencakup informasi apa pun yang kami inginkan untuk bertanya atau berkomunikasi dengan LLM, seperti instruksi bahasa alami dan konteks percakapan. Karena pendekatan ini tidak memerlukan pelatihan atau penyetelan tambahan, mengadaptasi LLM menjadi sangat efisien setelah dikembangkan. Hal ini agak mempengaruhi paradigma dalam NLP: kita tidak perlu lagi mengembangkan sistem khusus untuk tugas-tugas individual tetapi sebaliknya dapat menggunakan LLM tunggal yang terlatih dengan baik untuk melakukan tugas-tugas yang berbeda dengan mendorongnya. Aspek menarik dari dorongan LLM muncul sebagai hasil: pengguna dapat dengan mudah memiliki sistem yang “disesuaikan” dengan merancang perintah mereka sendiri untuk LLM. Mengingat peran penting yang dimainkan oleh prompting dalam LLM, rekayasa prompting telah menjadi bidang penelitian yang sangat aktif di NLP.

Istilah prompt digunakan dalam berbagai cara dalam literatur. Dalam bab ini, istilah ini mengacu pada seluruh masukan ke LLM, dan karenanya kita menggunakan istilah prompt dan model input secara bergantian. Sebelum membahas prompting lebih lanjut, mari kita lihat beberapa contoh di mana prompt lebih kompleks daripada yang disajikan pada subbagian sebelumnya. Perhatikan bahwa subbagian ini tidak ditujukan untuk menulis prompt berkualitas tinggi tetapi lebih pada menyoroti beberapa isu menarik di mendorong LLM.

Salah satu cara populer untuk menggunakan LLM adalah dengan menetapkan “peran” yang dimainkan dalam menghasilkan respons. Misalnya, LLM dapat bertindak sebagai psikolog saat menjawab pertanyaan.

Tolong jelaskan apa itu kepuasan yang tertunda?

Perlu dicatat, Anda adalah seorang peneliti dengan latar belakang yang mendalam di bidang psikologi. Bidang studi Anda sendiri adalah psikologi perkembangan pada anak-anak, dengan fokus pada aspek-aspek kontrol dan kecerdasan. Anda sangat familiar dengan konsep kepuasan yang tertunda dan telah menerapkannya pada penelitian Anda sendiri. Anda telah menerbitkan makalah tentang kepuasan yang tertunda dan melakukan eksperimen terkait.

Kepuasan yang tertunda adalah proses menolak hadiah langsung dengan harapan akan menerima hadiah yang lebih berharga di masa mendatang. Ini adalah konsep penting dalam psikologi perkembangan, terutama terkait dengan cara anak-anak belajar mengendalikan diri dan mengembangkan kemampuan kognitif. Penelitian menunjukkan bahwa kemampuan untuk menunda kepuasan dikaitkan dengan hasil positif dalam prestasi akademik, kompetensi sosial, dan kesejahteraan emosional. Studi dan eksperimen di bidang ini menyoroti pentingnya mengembangkan keterampilan ini sejak dini untuk mendukung keberhasilan dan ketahanan jangka panjang.

Contoh lain adalah penggunaan LLM dalam mendekripsi dan mengoreksi kesalahan seperti kesalahan sintaksis atau semantik dalam teks. Untuk LLM yang dilatih pada kode dan data bahasa alami, kita dapat menggunakan untuk debugging kode⁶.

Perbaiki bug dalam program bahasa C ini.

```
#include <stdio.h> int
main()
{
    print("Halo, Dunia!")
    return 0;
}
```

1. Nama fungsi 'printg' salah 2. Ada titik koma · Seharusnya 'printf'.
yang hilang di akhir pemanggilan fungsi 'printf'.

Berikut ini versi program yang sudah diperbaiki:

```
#include <stdio.h> int
main()
{
    printf("Halo, Dunia!");
    return 0;
}
```

Selain itu, LLM dapat disesuaikan untuk bertindak sebagai mitra percakapan⁷

⁶Dalam contoh ini, kode tidak diberi token agar lebih mudah dibaca.

⁷Untuk menyempurnakan LLM untuk dialog multi-giliran, seseorang perlu mempertimbangkan riwayat percakapan dalam konteks untuk memprediksi respons dalam putaran percakapan saat ini. Hal ini membuat perintah aktual yang digunakan dalam pembuatan respons relatif lebih panjang daripada yang digunakan dalam dialog satu putaran.

Hai! Saya asisten. Ada yang bisa saya bantu?

Siapa yang memenangkan Piala Dunia FIFA 2022?

Argentina memenangkan Piala Dunia FIFA 2022.

Dimana acaranya diadakan?

Piala Dunia FIFA 2022 diadakan di Qatar.

Berapa kali Argentina memenangkan Piala Dunia?

Argentina telah memenangkan Piala Dunia FIFA tiga kali.

...

Contoh-contoh ini dan contoh-contoh sebelumnya telah menunjukkan bahwa respons yang tepat dapat dihasilkan melalui perintah yang melibatkan instruksi dan pertanyaan yang jelas. Namun, ketika pemecahan masalah memerlukan pengetahuan yang tidak ditentukan secara eksplisit, LLM dapat membuat kesalahan, meskipun instruksinya cukup jelas dan tepat. Serangkaian tugas yang menantang untuk LLM melibatkan penalaran aritmatika dan penalaran akal sehat. Misalnya, kita dapat meminta LLM untuk memecahkan masalah matematika sekolah dasar yang disajikan dalam bahasa alami.

Jack punya 7 buah apel. Dia makan 2 buah apel untuk makan malam, tetapi ibunya memberinya 5 buah apel lagi. Keesokan harinya, Jack memberikan 3 buah apel kepada temannya, John. Berapa banyak apel yang tersisa pada Jack?

Jawabannya adalah 10.

Jawaban yang benar seharusnya 7, jadi keluaran modelnya salah.

Salah satu pendekatan untuk mengatasi masalah tersebut adalah dengan memasukkan pembelajaran ke dalam perintah, yang disebut pembelajaran dalam konteks atau (ICL). Ide ICL adalah untuk menunjukkan cara memecahkan masalah dalam perintah, dan prediksi kondisi pada demonstrasi ini. Berikut adalah contoh di mana masalah serupa dan jawaban yang sesuai disajikan dalam perintah (hijau = demonstrasi).

Tom memiliki 12 kelereng. Ia memenangkan 7 kelereng lagi dalam permainan dengan temannya tetapi kemudian kehilangan 5 kelereng keesokan harinya. Saudaranya memberinya 3 kelereng lagi sebagai hadiah. Berapa banyak kelereng yang dimiliki Tom sekarang?

Jawabannya adalah 17.

Jack punya 7 buah apel. Dia makan 2 buah apel untuk makan malam, tetapi ibunya memberinya 5 buah apel lagi. Keesokan harinya, Jack memberikan 3 buah apel kepada temannya, John. Berapa banyak apel yang tersisa pada Jack?

Jawabannya adalah 12.

Namun LLM masih membuat kesalahan kali ini. Salah satu alasannya mungkin karena penyelesaian masalah matematika tidak hanya melibatkan pemetaan masalah-jawaban, tetapi juga, pada tingkat yang lebih besar, inferensi logis yang mendasarinya dalam beberapa langkah. Salah satu metode untuk meningkatkan kemampuan inferensi LLM adalah dorongan berantai-pikiran (COT prompting) [Wei et al., 2022c]. Dalam dorongan COT, kami menguraikan masalah penalaran yang kompleks menjadi beberapa langkah perantara pemecahan masalah.

Langkah-langkah ditunjukkan dalam prompt sehingga LLM dapat diminta untuk belajar meniru proses berpikir yang mendasarinya dalam memecahkan masalah. Prompting COT telah ditemukan sangat berguna untuk mencapai hasil yang baik pada tugas-tugas yang menantang, seperti tolak ukur penalaran matematika GSM8K.

Perhatikan contoh demonstrasi di atas, Tom memiliki 12 kelereng. Ia memenangkan 7 kelereng lagi dalam permainan dengan temannya, tetapi kemudian kehilangan 5 kelereng keesokan harinya. Saudaranya memberinya 3 kelereng lagi sebagai hadiah. Berapa banyak kelereng yang dimiliki Tom sekarang? Langkah-langkah penalaran logis yang sesuai dapat disusun sebagai berikut:

1. Tom memulai dengan 12 kelereng.
2. Dia memenangkan 7 kelereng lagi dalam permainan, jadi kita menambahkannya ke hitungan awalnya: $12 \text{ kelereng} + 7 \text{ kelereng} = 19 \text{ kelereng}$.
3. Keesokan harinya, Tom kehilangan 5 kelereng, jadi kita kurangi 5 kelereng tersebut dari total kelerengnya saat ini: $19 \text{ kelereng} - 5 \text{ kelereng} = 14 \text{ kelereng}$.
4. Saudaranya memberinya 3 kelereng lagi, jadi kita tambahkan ke totalnya saat ini: $14 \text{ kelereng} + 3 \text{ kelereng} = 17 \text{ kelereng}$.
5. Setelah semua kejadian ini, Tom memiliki 17 kelereng.

Kita dapat menambahkan langkah-langkah penalaran ini ke dalam perintah, dan mendapatkan jawaban yang benar.

Tom memiliki 12 kelereng. Ia memenangkan 7 kelereng lagi dalam permainan dengan temannya tetapi kemudian kehilangan 5 kelereng keesokan harinya. Saudaranya memberinya 3 kelereng lagi sebagai hadiah. Berapa banyak kelereng yang dimiliki Tom sekarang?

Awalnya Tom memiliki 12 kelereng. Setelah memenangkan permainan, ia memperoleh 7 kelereng lagi, sehingga totalnya menjadi 19 kelereng. Ia kehilangan 5 kelereng keesokan harinya, sehingga totalnya menjadi 14 kelereng. Saudaranya memberinya 3 kelereng lagi, sehingga totalnya menjadi 17 kelereng. Jadi, Tom sekarang memiliki 17 kelereng. Jadi jawabannya adalah 17.

Jack punya 7 buah apel. Dia makan 2 buah apel untuk makan malam, tetapi ibunya memberinya 5 buah apel lagi. Keesokan harinya, Jack memberikan 3 buah apel kepada temannya, John. Berapa banyak apel yang tersisa pada Jack?

Jack mulai dengan 7 apel. Dia makan 2 apel untuk makan malam, jadi kita kurangi 2 dari 7, sehingga dia punya 5 apel. Ibunya memberinya 5 apel lagi, sehingga totalnya bertambah 5. Sekarang dia punya $5 + 5 = 10$ apel. Keesokan harinya, Jack memberi 3 apel kepada temannya John. Dengan mengurangi 3 dari 10, Jack punya 7 apel. Jadi, Jack punya 7 apel lagi. Jadi jawabannya adalah 7.

Pendekatan ini juga disebut one-shot COT prompting. Demikian pula, kita dapat menambahkan lebih banyak demonstrasi untuk mendapatkan pendekatan few-shot COT prompting.

Selain itu, jenis COT khusus, zero-shot COT, memperkenalkan instruksi yang memprovokasi LLM untuk berpikir, tanpa menghadirkan langkah penalaran perantara dalam perintah [Kojima et al.,

2022] Contoh menariknya adalah dengan menambahkan kalimat sederhana “Mari berpikir selangkah demi selangkah.” di akhir pertanyaan, LLM dapat menghasilkan langkah-langkah penalaran untuk mencapai jawaban akhir.

Jack punya 7 buah apel. Dia makan 2 buah apel untuk makan malam, tetapi ibunya memberinya 5 buah apel lagi. Keesokan harinya, Jack memberikan 3 buah apel kepada temannya, John. Berapa banyak apel yang tersisa pada Jack?

Mari kita berpikir selangkah demi selangkah.

1. Jumlah Awal: Jack mulai dengan 7 apel.
 2. Setelah Makan Malam: Dia makan 2 apel, jadi tersisa $7 - 2 = 5$ apel.
 3. Ibunya Memberi Lebih Banyak: Ibunya memberinya 5 apel lagi, jadi $5 + 5 = 10$ apel sekarang ada bersamanya.
 4. Memberikan kepada John: Keesokan harinya, Jack memberikan 3 buah apel kepada temannya, John, jadi tersisa $10 - 3 = 7$ apel.
- Pada akhirnya, Jack memiliki 7 apel tersisa.

Pembelajaran zero-shot, one-shot, dan few-shot merupakan konsep umum dalam area pembelajaran dalam konteks untuk LLM dan tidak terbatas pada prompting COT. Secara umum, prompting apa pun yang hanya melibatkan instruksi sederhana tanpa demonstrasi apa pun dapat dianggap sebagai bentuk pembelajaran zero-shot. Kemampuan pembelajaran zero-shot ini muncul saat LLM dilatih terlebih dahulu dan/atau disempurnakan. Selain itu, metode pembelajaran one-shot dan few-shot lebih sering dipertimbangkan saat LLM tidak memperoleh kemampuan pembelajaran zero-shot yang sesuai. Oleh karena itu, metode ini penting untuk pembelajaran dalam konteks saat menangani tugas baru. Contohnya termasuk yang melakukan berbagai tugas NLP dengan mendemonstrasikan sampel yang diformat tugas. Lihat contoh berikut untuk klasifikasi kalimat sentimen dan penerjemahan frasa melalui pembelajaran few-shot.

Berdasarkan potongan teks berikut, klasifikasikan sentimen mereka sebagai Positif, Negatif, atau Netral.

Contoh 1: “Saya menghabiskan hari yang luar biasa di taman!”

Sentimen: Positif

Contoh 2: “Pelayanan di restoran itu buruk.”

Sentimen: Negatif

Contoh 3: “Saya pikir hari ini akan hujan.”

Sentimen: Netral

Teks: “Film ini adalah perjalanan fantastis melalui imajinasi.”

Sentimen: Positif

Terjemahkan frasa bahasa Mandarin berikut ke bahasa Inggris.

Contoh 1: “ÿ”

Terjemahan: “Halo”

Contoh 2: “ÿÿ”

Terjemahan: “Terima kasih”

Frasa untuk menerjemahkan: “ÿÿ”

Terjemahan: “Selamat pagi”

Di atas, kami telah menyajikan contoh-contoh untuk mengilustrasikan kemampuan pembelajaran dalam konteks mendasar dalam LLM yang mendorong. Akan tetapi, bagian ini tidak menyertakan teknik-teknik mendorong yang lebih canggih untuk menjaga konten tetap ringkas dan padat. Pembahasan lebih lanjut tentang mendorong dapat ditemukan di Bab 3.

2.2 Pelatihan dalam Skala Besar

Sebagai langkah pertama dalam mengembangkan LLM, kita perlu melatih model ini pada sejumlah besar data. Tugas pelatihan itu sendiri standar: tujuannya adalah untuk memaksimalkan kemungkinan, yang dapat dicapai melalui penurunan gradien. Namun, saat kita meningkatkan ukuran model dan jumlah data, masalahnya menjadi sangat menantang, misalnya, model besar umumnya membuat pelatihan tidak stabil. Di bagian ini, kami membahas beberapa masalah pelatihan skala besar untuk LLM, termasuk persiapan data, modifikasi model, dan pelatihan terdistribusi. Kami juga membahas hukum penskalaan untuk LLM, yang membantu kita memahami efisiensi dan efektivitas pelatihannya.

2.2.1 Persiapan Data

Pentingnya data tidak dapat dilebih-lebihkan dalam NLP. Seiring dengan berkembangnya jaringan saraf yang lebih besar, permintaan akan data terus meningkat. Misalnya, pengembangan LLM mungkin memerlukan triliunan token dalam pra-pelatihan (lihat Tabel 2.3), yang jumlahnya jauh lebih besar daripada yang digunakan dalam pelatihan model NLP konvensional. Secara umum, kita mungkin ingin mengumpulkan data pelatihan sebanyak mungkin. Namun, kumpulan data pelatihan yang lebih besar tidak berarti hasil pelatihan yang lebih baik, dan pengembangan LLM menimbulkan masalah baru dalam pembuatan atau pengumpulan kumpulan data ini.

Masalah pertama adalah kualitas data. Data berkualitas tinggi telah lama dianggap penting untuk melatih sistem NLP berbasis data. Menggunakan teks mentah dari berbagai sumber secara langsung pada umumnya tidak diinginkan. Misalnya, sebagian besar data yang digunakan untuk melatih LLM terbaru berasal dari web scraping, yang mungkin berisi kesalahan dan konten yang tidak pantas, seperti informasi yang beracun dan fakta yang dibuat-buat. Selain itu, internet dibanjiri dengan konten yang dihasilkan mesin karena meluasnya penggunaan AI, yang menghadirkan tantangan lebih lanjut untuk memproses dan menggunakan data yang diambil dari web. Para peneliti telah menemukan bahwa melatih LLM pada data yang tidak difilter berbahaya [Raffel et al., 2020]. Meningkatkan kualitas data biasanya melibatkan penggabungan langkah-langkah penyaringan dan pembersihan dalam alur kerja pemrosesan data. Misalnya, Penedo et al. [2023] menunjukkan bahwa dengan mengadopsi sejumlah teknik pemrosesan data, 90% dari data yang diambil dari web dapat dihapus untuk pelatihan LLM. Selain data yang diambil dari web dalam skala besar, data pelatihan LLM sering kali mencakup buku, makalah, data yang dibuat pengguna di media sosial, dan sebagainya. Sebagian besar LLM terbaru dilatih pada kumpulan data gabungan tersebut, yang ditemukan

2.2 Pelatihan di Skala

57

Magister Hukum	# Data Token
GPT3-175B [Brown dkk., 2020]	0,5T Halaman Web, Buku, Wikipedia
Falcon-180B [Almazrouei dkk., 2023]	3.5T Halaman Web, Buku, Percakapan, Kode, Artikel Teknis
LLaMA2-65B [Touvron et al., 2023a]	1.0T ÷ 1.4T Halaman Web, Kode, Wikipedia, Buku, Makalah, Tanya Jawab
PaLM-450B [Chowdhery dkk., 2022]	0,78T Halaman Web, Buku, Percakapan, Kode, Wikipedia, Berita
Gemma-7B [Tim Gemma, 2024]	Halaman Web 6T, Matematika, Kode

Tabel 2.3: Jumlah data pelatihan yang digunakan dalam beberapa LLM dalam hal jumlah token.

penting untuk kinerja yang kuat dari model yang dihasilkan.

Masalah kedua adalah keberagaman data. Kami ingin data pelatihan mencakup sebanyak mungkin jenis data. data sebanyak mungkin, sehingga model yang dilatih dapat beradaptasi dengan berbagai tugas hilir dengan mudah. Ini memiliki telah diakui secara luas bahwa kualitas dan keragaman data pelatihan memainkan peran yang sangat penting peran dalam LLM. Contoh menariknya adalah menggabungkan kode pemrograman ke dalam data pelatihan telah ditemukan bermanfaat bagi LLM. Manfaatnya ditunjukkan tidak hanya dalam meningkatkan kemampuan pemrograman LLM, tetapi juga dalam meningkatkan penalaran untuk masalah yang kompleks, terutama mereka yang membutuhkan dorongan COT. Konsep “keanekaragaman” dapat diperluas untuk mencakup bahasa keragaman juga. Misalnya, banyak LLM dilatih pada data multibahasa, dan karena itu kami dapat menangani beberapa bahasa menggunakan satu model. Meskipun pendekatan ini menunjukkan kemampuan yang kuat dalam tugas multibahasa dan lintas bahasa, kinerjanya pada bahasa tertentu sangat bergantung pada volume dan kualitas data untuk bahasa-bahasa tersebut. Dalam beberapa kasus telah terbukti memberikan hasil yang buruk untuk bahasa dengan sumber daya rendah.

Masalah ketiga adalah bias dalam data pelatihan. Ini bukan masalah yang khusus untuk LLM, tetapi terdapat dalam banyak sistem NLP. Contoh umum adalah bias gender, dimana LLM menunjukkan preferensi untuk satu jenis kelamin dibandingkan dengan jenis kelamin lainnya. Hal ini sebagian dapat dikaitkan dengan ketidakseimbangan kelas dalam data pelatihan, misalnya, istilah perawat lebih sering dikaitkan dengan wanita. Untuk menghilangkan bias data, merupakan praktik umum untuk menyeimbangkan kategori fenomena bahasa yang berbeda, seperti gender, etnis, dan dialek. Bias dalam data juga terkait dengan masalah keberagaman yang disebutkan di atas. Misalnya, karena banyak LLM yang dilatih dan disesuaikan dengan data yang berpusat pada bahasa Inggris, mereka bias terhadap nilai-nilai dan perspektif budaya yang berlaku di antara populasi penutur bahasa Inggris. Meningkatkan keberagaman bahasa pada data pelatihan dapat mengurangi bias.

Masalah lain dalam pengumpulan data berskala besar adalah masalah privasi. Jika LLM dilatih berdasarkan data dari sumber yang luas, hal ini berpotensi menimbulkan risiko mengenai paparan informasi sensitif informasi, seperti kekayaan intelektual dan data pribadi. Hal ini khususnya menjadi perhatian mengingat kapasitas LLM untuk merepresentasikan pola dari data yang mereka gunakan untuk pelatihan, yang mungkin secara tidak sengaja melibatkan menghafal dan mereproduksi detail tertentu. Pendekatan sederhana terhadap privasi perlindungan adalah untuk menghapus atau menganonimkan informasi sensitif. Misalnya, teknik anonimisasi dapat diterapkan untuk menghapus informasi identitas pribadi dari data pelatihan untuk mencegah LLM dari pembelajaran dari data tersebut. Namun, dalam praktiknya, menghapus atau menyunting semua data sensitif sulit. Oleh karena itu, banyak LLM, terutama yang diluncurkan untuk layanan publik, biasanya bekerja dengan sistem yang dapat mendeteksi potensi paparan data sensitif, atau disetel dengan baik untuk menolak

permintaan tertentu yang dapat menyebabkan kebocoran informasi.

2.2.2 Modifikasi Model

Pelatihan LLM sulit. Masalah yang sering ditemui adalah bahwa proses pelatihan menjadi lebih tidak stabil saat LLM bertambah besar. Misalnya, seseorang perlu memilih laju pembelajaran yang kecil untuk mencapai pelatihan yang stabil dengan penurunan gradien, tetapi hal ini pada gilirannya menghasilkan waktu pelatihan yang jauh lebih lama. Terkadang, bahkan ketika konfigurasi pelatihan dirancang dengan cermat, pelatihan dapat menyimpang pada titik-titik tertentu selama pengoptimalan. Pelatihan LLM umumnya dipengaruhi oleh banyak faktor, seperti inisialisasi parameter, batching, dan regularisasi. Di sini, kami fokus pada modifikasi dan peningkatan umum pada arsitektur Transformer standar, yang dianggap penting dalam mengembangkan LLM yang dapat dilatih.

2.2.2.1 Normalisasi Lapisan dengan Koneksi Residual

Normalisasi lapisan digunakan untuk menstabilkan pelatihan untuk jaringan saraf dalam. Proses ini dilakukan dengan mengurangi nilai rata-rata dan membaginya dengan deviasi standar. Dengan menormalkan keluaran lapisan dengan cara ini, kita dapat secara efektif mengurangi masalah pergeseran kovariat dan meningkatkan stabilitas p. Dalam Transformer, normalisasi lapisan biasanya digunakan bersama dengan koneksi residual. Seperti yang dijelaskan dalam Bagian 2.1.1, sublapisan dapat didasarkan pada arsitektur post-norm, di mana normalisasi lapisan dilakukan tepat setelah blok residual, atau arsitektur pre-norm, di mana normalisasi lapisan dilakukan di dalam blok residual. Sementara kedua arsitektur ini banyak digunakan dalam sistem berbasis Transformer [Wang et al., 2019], arsitektur pre-norm telah terbukti sangat berguna dalam melatih Transformer yang mendalam. Mengingat hal ini, sebagian besar LLM didasarkan pada arsitektur pre-norm, yang dinyatakan sebagai output = $\text{LN}(F(\text{input})) + \text{input}$.

Bentuk fungsi normalisasi lapisan yang banyak digunakan diberikan oleh

$$\text{Norma } L(h) = \frac{h - \mu}{\sigma} \quad (2.23)$$

di mana h adalah vektor bernilai riil berdimensi d , μ adalah rata-rata semua entri h , dan σ adalah deviasi standar yang sesuai dengan d . σ diperkenalkan demi stabilitas numerik. $\sigma \in \mathbb{R}$ dan $\sigma \neq 0$

D adalah istilah gain dan bias.

Varian normalisasi lapisan, disebut normalisasi lapisan root mean square (RMS), hanya menskalakan ulang vektor input tetapi tidak memusatkannya kembali [Zhang dan Sennrich, 2019]. Fungsi normalisasi lapisan RMS diberikan oleh

$$\text{LN}(h) = \frac{h}{\sqrt{\frac{1}{D} \sum_{i=1}^D h_i^2}} \quad (2.24)$$

di mana $\sqrt{\frac{1}{D} \sum_{i=1}^D h_i^2}$ adalah akar rata-rata kuadrat dari h , yaitu, fungsi $\sqrt{\frac{1}{D} \sum_{i=1}^D h_i^2}$. Normalisasi lapisan ini $\sqrt{\frac{1}{D} \sum_{i=1}^D h_i^2}$ digunakan dalam LLM seperti seri LLaMA.

2.2.2.2 Fungsi Aktivasi dalam FFN

Dalam Transformers, sub-lapisan FFN dirancang untuk memperkenalkan non-linieritas ke dalam pembelajaran representasi, dan ditemukan berguna untuk mencegah representasi yang dipelajari dengan perhatian diri dari degenerasi⁸ [Dong et al., 2021]. Bentuk standar FFN yang digunakan dalam sub-lapisan ini dapat dinyatakan sebagai

$$\text{FFN}(h) = \tilde{y}(hWh + bh)Wf + bf \quad (2.25)$$

di mana $Wh \in \mathbb{R}^{d \times dh}$, $bh \in \mathbb{R}^{dh}$, $Wf \in \mathbb{R}^{dh \times d}$ adalah parameter, dan dh adalah ukuran tersembunyi. $\tilde{y}(\cdot)$ adalah fungsi aktivasi dari lapisan tersembunyi. Pilihan umum untuk $\tilde{y}(\cdot)$ adalah unit linier yang diperbaiki (ReLU), yang diberikan oleh

$$\tilde{y}_{\text{relu}}(h) = \max(0, h) \quad (2.26)$$

Dalam implementasi praktis, peningkatan dh sangat membantu dan karenanya sering ditetapkan ke angka yang lebih besar dalam LLM. Namun, ukuran tersembunyi yang sangat besar menimbulkan tantangan untuk pelatihan dan penerapan. Dalam kasus ini, desain fungsi aktivasi memainkan peran yang relatif lebih penting dalam FFN yang ada. Ada beberapa alternatif untuk ReLU dalam LLM. Salah satunya adalah unit linier gaussian error (GeLU) yang dapat dilihat sebagai versi ReLU yang dihaluskan. Daripada mengendalikan output berdasarkan tanda input, fungsi GeLU memberi bobot pada inputnya berdasarkan persentil $\Pr(h \leq h)$.

Di sini h adalah vektor d -dimensi yang entri-entrinya diambil dari distribusi normal standar

$\text{Gaussian}(0, 1)$ ⁹. Secara khusus, fungsi GeLU didefinisikan sebagai

$$\tilde{y}_{\text{gelu}}(h) = h \Pr(h \leq h) = h \tilde{y}(h) \quad (2.27)$$

di mana $\tilde{y}(h)$ adalah fungsi distribusi kumulatif Gaussian($0, 1$), yang dapat diimplementasikan dengan cara yang mudah [Hendrycks dan Gimpel, 2016]. Fungsi GeLU telah diadopsi dalam beberapa LLM, seperti BERT, GPT-3, dan BLOOM.

Keluarga fungsi aktivasi lain yang populer dalam LLM adalah fungsi berbasis unit linier berpagar (GLU). Bentuk dasar GLU diberikan oleh

$$\tilde{y}_{\text{glu}}(h) = \tilde{y}(hW_1 + b_1) \tilde{y}(W_2 + b_2) \quad (2.28)$$

dimana $W_1 \in \mathbb{R}^{d \times d}$, $b_1 \in \mathbb{R}^d$, $W_2 \in \mathbb{R}^{d \times d}$, dan $b_2 \in \mathbb{R}^d$ adalah parameter model. Pilihan $\tilde{y}(\cdot)$ yang berbeda menghasilkan versi fungsi GLU yang berbeda. Misalnya, jika $\tilde{y}(\cdot)$ didefinisikan sebagai fungsi GeLU, kita akan memiliki fungsi GeGLU

$$\tilde{y}_{\text{gelu}}(h) = \tilde{y}_{\text{gelu}}(hW_1 + b_1) \tilde{y}(W_2 + b_2) \quad (2.29)$$

Fungsi aktivasi ini telah berhasil diterapkan di LLM seperti Gemma.

Sebagai contoh lain, pertimbangkan $\tilde{y}(\cdot)$ sebagai fungsi Swish $\tilde{y}_{\text{swish}}(h) = h \tilde{y}_{\text{sigmoid}}(ch)$

⁸Di sini degenerasi mengacu pada fenomena di mana peringkat matriks dikurangi setelah beberapa pemrosesan.

⁹ $\Pr(h \leq h)$ adalah notasi informal. Ini mengacu pada vektor, dengan setiap entri mewakili persentil untuk entri yang sesuai dari h .

[Ramachandran et al., 2017]. Kemudian, fungsi SwiGLU diberikan oleh

$$\hat{y}_{\text{swiglu}}(h) = \hat{y}_{\text{swish}}(hW_1 + b_1) \hat{y}(W_2 + b_2) \quad (2.30)$$

Baik seri PaLM dan LLaMA didasarkan pada fungsi SwiGLU. Untuk diskusi lebih lanjut GLU, pembaca dapat merujuk pada karya Shazeer [2020].

2.2.2.3 Menghapus Istilah Bias

Desain model populer lainnya adalah menghapus istilah bias dalam transformasi afin yang digunakan dalam LLM. Perlakuan ini dapat diterapkan pada normalisasi lapisan, transformasi masukan untuk perhatian QKV, dan FFN. Misalnya, kita dapat memodifikasi Persamaan (2.25) untuk memperoleh FFN tanpa bias.

$$\text{FFN}(h) = \hat{y}(hWh)Wf \quad (2.31)$$

Chowdhery et al. [2022] melaporkan bahwa menghilangkan istilah bias membantu meningkatkan stabilitas pelatihan LLM. Metode ini telah digunakan dalam beberapa LLM terkini, seperti LLaMA dan Gemma.

2.2.2.4 Masalah Lainnya

Banyak LLM juga melibatkan modifikasi pada model penempatan posisionalnya. Misalnya, seseorang dapat mengganti penyandian posisional sinusoidal dengan penempatan posisi putar sehingga LLM yang dipelajari dapat menangani urutan panjang dengan lebih baik. Model-model ini akan dibahas di Bagian 2.3.

Perhatikan bahwa meskipun modifikasi model umum dilakukan dalam pelatihan LLM, stabilitas pelatihan dapat ditingkatkan dengan berbagai cara. Misalnya, meningkatkan ukuran batch saat pelatihan berlangsung telah terbukti bermanfaat untuk beberapa LLM. Secara umum, mencapai pelatihan LLM skala besar yang stabil dan efisien memerlukan pengaturan yang dirancang dengan cermat, termasuk jadwal pembelajaran, pilihan pengoptimal, paralelisme pelatihan, pelatihan presisi campuran, dan sebagainya. Beberapa masalah ini sangat direkayasa, dan oleh karena itu, kita biasanya memerlukan sejumlah pelatihan untuk mendapatkan LLM yang memuaskan.

2.2.3 Pelatihan Terdistribusi

Pelatihan LLM memerlukan sumber daya komputasi yang signifikan. Pendekatan umum untuk meningkatkan efisiensi pelatihan adalah dengan menggunakan sistem terdistribusi berskala besar. Untungnya, seiring dengan munculnya jaringan saraf dalam AI, perangkat lunak dan perangkat keras yang berorientasi pada pembelajaran mendalam telah dikembangkan, sehingga memudahkan penerapan LLM dan melakukan komputasi. Misalnya, seseorang sekarang dapat dengan mudah menyempurnakan LLM menggunakan kerangka kerja perangkat lunak pembelajaran mendalam. Namun, peningkatan pelatihan LLM masih menantang, dan memerlukan upaya signifikan dalam mengembangkan sistem perangkat keras dan perangkat lunak untuk pelatihan terdistribusi yang stabil dan efisien.

Pertimbangan penting dari pelatihan terdistribusi adalah paralelisme. Ada beberapa bentuk paralelisme: paralelisme data, paralelisme model, paralelisme tensor, dan paralelisme jalur pipa.

Meskipun ada berbagai cara untuk mendistribusikan komputasi di seluruh perangkat, metode paralelisme ini didasarkan pada ide yang sama: masalah pelatihan dapat dibagi menjadi tugas-tugas yang lebih kecil yang dapat dieksekusi secara bersamaan. Masalah paralelisme dalam pelatihan LLM telah dipelajari secara ekstensif.

[Narayanan dkk., 2021; Fedus dkk., 2022]. Di sini kita membuat sketsa konsep dasar.

- **Paralelisme Data.** Metode ini merupakan salah satu metode paralelisme yang paling banyak digunakan untuk melatih jaringan saraf. Sebagai ilustrasi, perhatikan kasus paling sederhana di mana aturan delta standar digunakan dalam penurunan gradien.

$$\hat{y}_{t+1} = \hat{y}_t - lr \cdot \hat{y}'_t \quad \underline{\hat{y}'_t \text{ (Dmini)}} \quad (2.32)$$

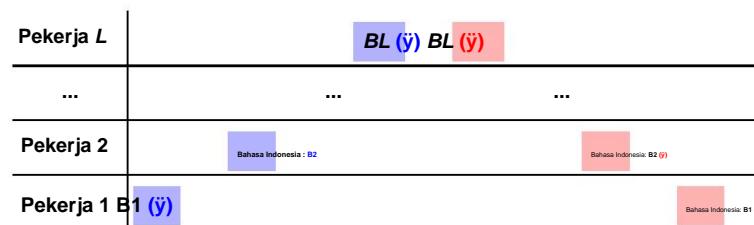
di mana parameter baru \hat{y}_{t+1} diperoleh dengan memperbarui parameter terbaru \hat{y}_t dengan langkah \hat{y}'_t (Dmini) kecil lr dalam arah gradien kerugian negatif. adalah gradien kerugian \hat{y}'_t terhadap parameter y_t , dan dihitung pada minibatch sampel pelatihan Dmini.

Dalam paralelisme data, kami membagi Dmini menjadi N kelompok yang lebih kecil, dilambangkan dengan $\{D_1, \dots, D_N\}$. Kemudian, kami mendistribusikan kelompok-kelompok ini ke N pekerja, masing-masing dengan kelompok yang sesuai. Setelah data didistribusikan, para pekerja ini dapat bekerja pada waktu yang sama. Gradien seluruh kelompok mini diperoleh dengan menggabungkan gradien yang dihitung oleh para pekerja, seperti ini

$$\underline{\hat{y}'_t \text{ (Dmini)}} = \frac{\text{Tentukan } \hat{y}'_t \text{ (D1)} + \hat{y}'_t \text{ (D2)} + \dots + \hat{y}'_t \text{ (D}_N\text{)}}{\text{pekerja 1} \quad \text{pekerja 2} \quad \text{pekerja } N} \quad (2.33)$$

Dalam kasus ideal di mana para pekerja berkoordinasi dengan baik dan overhead komunikasi kecil, paralelisme data dapat mencapai kecepatan pelatihan hampir N kali lipat.

- **Paralelisme Model.** Meskipun paralelisme data sederhana dan efektif, hal ini mengharuskan setiap pekerja untuk menjalankan seluruh LLM dan melakukan proses maju dan mundur secara lengkap. Bahasa Indonesia: Seiring LLM tumbuh lebih besar, terkadang menjadi tidak layak untuk memuat dan menjalankan LLM pada satu perangkat. Dalam hal ini, kita dapat memisahkan LLM menjadi komponen yang lebih kecil dan menjalankan komponen ini pada perangkat yang berbeda. Salah satu cara sederhana untuk melakukannya adalah dengan mengelompokkan lapisan yang berurutan dalam tumpukan lapisan dan menetapkan setiap kelompok ke pekerja. Pekerja beroperasi dalam urutan lapisan dalam tumpukan, yaitu, dalam lintasan maju kita memproses input dari lapisan tingkat bawah ke lapisan tingkat atas, dan dalam lintasan mundur kita menyebarkan gradien kesalahan dari lapisan tingkat atas ke lapisan tingkat bawah. Pertimbangkan, misalnya, dekoder Transformer dengan blok bertumpuk L . Untuk mendistribusikan beban komputasi, setiap blok ditetapkan ke pekerja. Lihat ilustrasi berikut untuk satu k



Di sini B menunjukkan perhitungan blok l , dan simbol \hat{y} dan \check{y} menunjukkan lintasan maju dan mundur, masing-masing. Perhatikan bahwa metode paralelisme ini memaksa pekerja untuk bekerja secara berurutan, sehingga pekerja harus menunggu pekerja sebelumnya menyelesaikan pekerjaannya. Hal ini mengakibatkan perangkat tidak digunakan selama sebagian besar waktu. Dalam sistem praktis, paralelisme model umumnya digunakan bersama dengan mekanisme paralelisme lainnya untuk memaksimalkan penggunaan

- **Paralelisme Tensor.** Paralelisme juga dapat dilakukan dalam satu langkah komputasi. Contoh umum adalah membagi matriks parameter besar menjadi beberapa bagian, mengalikan tensor input dengan masing-masing bagian ini secara terpisah, lalu menggabungkan hasil perkalian ini untuk membentuk output. Misalnya, perhatikan perkalian repre-dengan matriks parameter W_h yang kalimat $h \in \mathbb{R}^{d \times d_h}$ dalam sublapisan FFN (lihat Pers. (2.25)). Kita dapat mengiris matriks $W_h \in \mathbb{R}^{d \times d_h}$ secara vertikal menjadi urutan M sub-matriks

$$A_p = W_1 \quad H \quad W_2 \text{ jam } \dots \text{ } W_M \quad H \quad (2.34)$$

dimana setiap sub-matriks $W_k \in \mathbb{R}^{d_h \times M}$ memiliki bentuk $d \times M$. Perkalian h dengan W_h dapat dilakukan dinyatakan sebagai

$$\begin{aligned} hW_h &= hW_1 \quad H \quad W_2 \text{ jam } \dots \text{ } W_M \quad H \\ &= hW_1 \quad H \quad hW_2 \quad h \dots \text{ } hW_M \quad H \end{aligned} \quad (2.35)$$

Kita dapat melakukan perkalian matriks $\{hW_1, hW_2, \dots, hW_M\}$ pada perangkat M secara terpisah. Hasilnya, kita mendistribusikan perkalian matriks yang besar di beberapa perangkat, yang masing-masing mungkin memiliki memori yang relatif kecil. Dari perspektif desain GPU modern, paralelisme tensor pada GPU menyediakan pendekatan berbasis petak dua tingkat untuk komputasi paralel. Pertama, pada tingkat yang lebih tinggi, kita menguraikan perkalian matriks menjadi perkalian sub-matriks yang dapat langsung masuk ke dalam memori GPU. Kemudian, pada tingkat yang lebih rendah, kita menjalankan perkalian sub-matriks ini pada GPU menggunakan algoritme paralel berbasis petak yang secara khusus dioptimalkan untuk GPU.

- **Paralelisme Pipeline.** Di atas, dalam paralelisme model, kami telah menjelaskan pendekatan sederhana untuk menyebarluaskan kelompok komponen model ke beberapa perangkat. Namun, metode ini tidak efisien karena hanya satu perangkat yang diaktifkan pada satu waktu selama pemrosesan. Paralelisme pipeline mengatasi masalah ini dengan memperkenalkan tumpang tindih antara komputasi pada perangkat yang berbeda [Harlap et al., 2018; Huang et al., 2019]. Untuk melakukan ini, sekumpulan sampel dibagi menjadi sejumlah mikro-batch, lalu mikro-batch ini diproses oleh setiap pekerja seperti biasa. Setelah mikro-batch diproses oleh seorang pekerja dan diteruskan ke pekerja berikutnya, mikro-batch berikutnya segera menempati pekerja yang sama. Dengan kata lain, kami membuat pipeline di mana langkah-langkah komputasi yang berbeda dapat tumpang tindih jika beberapa pekerjaan diberikan ke pipeline. Berikut ini menunjukkan ilustrasi paralelisme pipeline untuk memproses 3 mikro-batch.

Pekerja L	$BL,1$ $BL,2$ $BL,3$ $BL,1$ $BL,2$ $BL,3$
...	...
Pekerja 2	$B2,1$ $B2,2$ $B2,3$
Pekerja 1	$B1,1$ $B1,2$ $B1,3$

Di sini BL,k mewakili pemrosesan mikro-batch ke- k oleh pekerja ke- l . Idealnya kita ingin memaksimalkan jumlah mikro-batch, dan dengan demikian meminimalkan waktu menganggur

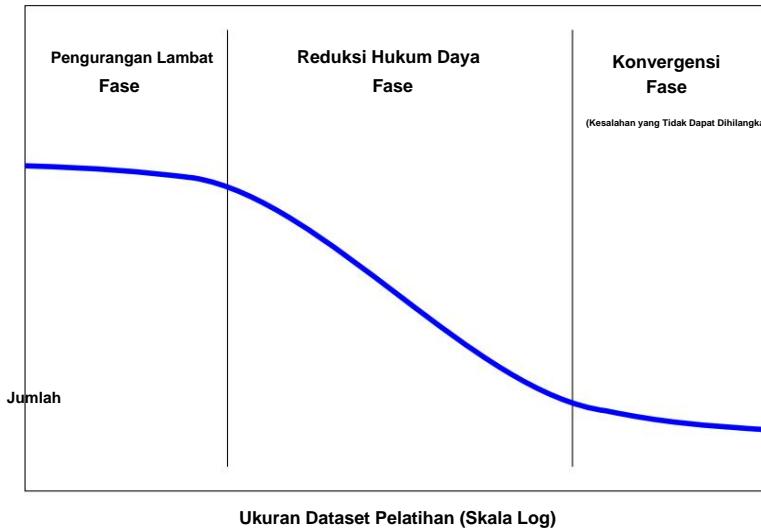
pekerja. Namun, dalam praktiknya, penggunaan mikro-batch kecil sering kali mengurangi pemanfaatan GPU dan meningkatkan biaya pengalihan tugas. Hal ini pada gilirannya dapat menurunkan throughput sistem secara keseluruhan.

Tujuan akhir dari pemrosesan paralel adalah untuk mencapai pertumbuhan linier dalam efisiensi, yaitu jumlah sampel yang dapat diproses per satuan waktu meningkat secara linier seiring dengan jumlah perangkat. Namun, pelatihan terdistribusi rumit, dan dipengaruhi oleh banyak faktor selain dengan metode parallelisme yang kita pilih. Salah satu masalah yang sering dikaitkan dengan distribusi sistem, adalah biaya komunikasi. Kita dapat menganggap sistem terdistribusi sebagai sekelompok node yang terhubung dalam jaringan. Setiap node ini dapat melakukan komputasi lokal atau meneruskan data ke node lain. Jika ada sejumlah besar node seperti itu, akan mahal untuk mendistribusikan dan mengumpulkan data di seluruh. Terkadang, penghematan waktu yang disebabkan oleh parallelisme diimbangi oleh overhead komunikasi jaringan yang besar. Masalah lain dengan sistem terdistribusi skala besar adalah sinkronisasi node menimbulkan biaya tambahan. Seperti yang sering terjadi, beberapa node mungkin membutuhkan waktu lebih lama untuk bekerja, menyebabkan yang lain menunggu yang paling lambat. Sementara kita dapat menggunakan asynchronous pelatihan untuk menangani heterogenitas dalam sumber daya komputasi, hal ini dapat menyebabkan gradien basi dan konvergensi yang tidak terjamin. Selain itu, karena semakin banyak node yang ditambahkan ke jaringan, maka akan semakin banyak kemungkinan node mengalami crash selama pelatihan. Dalam kasus ini, kita perlu memastikan bahwa keseluruhan sistem toleran terhadap kesalahan. Dalam banyak pengaturan praktis, untuk meningkatkan skalabilitas, seseorang perlu mempertimbangkan memperhitungkan masalah tambahan, termasuk desain arsitektur, transfer data dan tumpang tindih komputasi, penyeimbangan beban, lebar pita memori, dan seterusnya.

Pelatihan LLM sangat mahal secara komputasi sehingga, meskipun pelatihan terdistribusi sudah digunakan, para peneliti dan insinyur sering kali masih menggunakan berbagai metode kompresi dan percepatan model untuk meningkatkan efisiensi pelatihan [Weng, 2021]. Salah satu contohnya adalah pelatihan presisi campuran, di mana data presisi rendah (seperti data FP16 dan FP8) digunakan untuk perhitungan gradien pada setiap node individual, dan data presisi tunggal atau ganda (seperti data FP32/FP64) digunakan untuk memperbarui model [Micikevicius et al., 2018]. Operasi kunci dalam pendekatan ini adalah akumulasi gradien di mana gradien perlu diakumulasikan dan disinkronkan di seluruh node. Namun, karena non-asosiatif penjumlahan floating-point, hal ini dapat menyebabkan sedikit perbedaan numerik dalam gradien terakumulasi pada node yang berbeda, yang dapat memengaruhi konvergensi model dan kinerja akhir. Masalah ini lebih jelas jika ada sejumlah besar node yang terlibat dalam pelatihan terdistribusi, terutama mengingat bahwa komputasi numerik dengan presisi rendah mungkin mengalami masalah luapan dan kekurangan aliran, serta ketidakkonsistenan di berbagai perangkat keras. Oleh karena itu, desain Sistem terdistribusi perlu mempertimbangkan masalah komputasi numerik ini untuk memastikan kepuasan hasil dan konvergensi.

2.2.4 Hukum Skala

Keberhasilan LLM menunjukkan bahwa melatih model bahasa yang lebih besar dengan menggunakan lebih banyak sumber daya dapat menghasilkan untuk meningkatkan kinerja model. Para peneliti telah menjelaskan hal ini sebagai hukum penskalaan LLM. Selengkapnya Secara khusus, hukum skala menggambarkan hubungan antara kinerja LLM dan atribut pelatihan LLM, seperti ukuran model, jumlah komputasi yang digunakan untuk pelatihan, dan jumlah data pelatihan. Misalnya, Hestness et al. [2017] menunjukkan bahwa kinerja jaringan saraf dalam adalah fungsi seperti hukum pangkat dari ukuran data pelatihan. Pada awalnya, ketika jumlah data pelatihan tidak besar, kinerja model membaik secara perlahan. Setelah itu, ketika lebih banyak data pelatihan digunakan, model memasuki fase peningkatan kinerja yang cepat, dan kurva kinerja menyerupai kurva hukum daya. Pada akhirnya, peningkatan kinerja



Gambar 2.3: Hukum penskalaan kesalahan pengujian terhadap variabel yang diinginkan (misalnya, ukuran set data pelatihan) [Hestness et al., 2017]. Kurva hukum penskalaan dapat dibagi menjadi tiga fase. Pada awalnya, jumlah kesalahan pengujian berkurang secara perlahan ketika lebih banyak data pelatihan digunakan, tetapi ini hanya berlangsung dalam waktu singkat. Pada fase kedua, jumlah kesalahan pengujian berkurang drastis, dan kurva menjadi kurva hukum pangkat. Setelah itu, pengurangan kesalahan melambat lagi pada fase ketiga. Perhatikan bahwa ada kesalahan yang tidak dapat dikurangi yang tidak dapat dihilangkan, terlepas

menjadi lambat lagi, dan lebih banyak data tidak menghasilkan keuntungan yang signifikan. Gambar 2.3 menunjukkan contoh kurva tersebut.

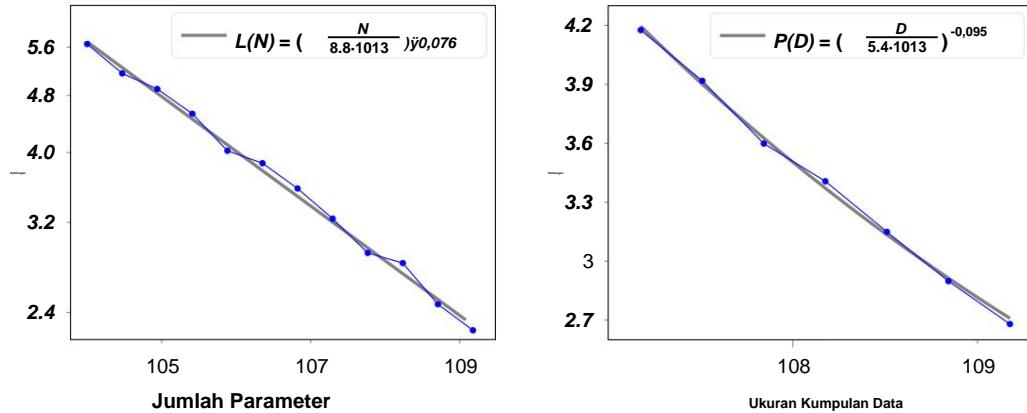
Dalam NLP, pandangan tradisional menyatakan bahwa peningkatan performa akan hilang pada titik tertentu saat pelatihan ditingkatkan. Namun, hasil terbaru menunjukkan bahwa, jika kita mempertimbangkan masalah dalam skala yang lebih besar, peningkatan pelatihan masih merupakan metode yang sangat efektif untuk memperoleh LLM yang lebih baik. Misalnya, LLM sumber tertutup dan sumber terbuka dapat memperoleh manfaat dari lebih banyak data, meskipun triliunan token telah digunakan untuk pelatihan.

Dengan peningkatan skala pelatihan model, LLM menunjukkan kapabilitas baru, yang dikenal sebagai kemampuan LLM yang muncul. Misalnya, Wei et al. [2022b] mempelajari properti penskalaan LLM di berbagai ukuran model dan jumlah sumber daya komputasi. Pekerjaan mereka menunjukkan bahwa beberapa kemampuan muncul ketika kita menskalakan ukuran model ke tingkat tertentu. Munculnya kemampuan yang muncul telah menunjukkan peran pelatihan berskala dalam meningkatkan kinerja LLM, dan juga, sampai batas tertentu, memotivasi para peneliti untuk terus mencoba melatih model yang lebih besar. Karena LM yang lebih besar dan lebih kuat terus muncul, pemahaman kita tentang hukum penskalaan terus matang. Ini membantu para peneliti memprediksi kinerja LLM selama pelatihan dan memperkirakan sumber daya komputasi minimal yang diperlukan untuk mencapai tingkat kinerja tertentu.

Untuk memahami bagaimana kinerja model berskala dengan berbagai faktor yang dipertimbangkan selama pelatihan, kinerja model biasanya dinyatakan sebagai fungsi dari faktor-faktor ini. Misalnya, dalam kasus yang paling sederhana, kita dapat menyatakan kerugian atau kesalahan LLM sebagai fungsi dari satu variabel yang diminati. Namun, tidak ada hukum penskalaan universal yang dapat menjelaskan hubungan ini. Sebaliknya, fungsi yang berbeda diusulkan agar sesuai dengan kurva pembelajaran LLM.

Misalkan x adalah variabel yang diinginkan (seperti jumlah parameter model) dan $L(x)$ adalah kerugian model yang diberikan x (seperti kerugian entropi silang pada data uji). Bentuk paling sederhana dari $L(x)$ adalah hukum pangkat

$$L(x) = ax^b \quad (2.36)$$



Gbr. 2.4: Pengujian kerugian terhadap ukuran model (N) dan ukuran kumpulan data pelatihan (D) (titik data diplot untuk tujuan ilustrasi).

Kami memplot kerugian pengujian sebagai fungsi N, yang didefinisikan sebagai $\frac{N}{8,8 \times 10^{13}}$ -0,076, dan fungsi D, yaitu $L(N) = \text{didefinisikan sebagai } L(\theta) = \frac{\theta}{5,4 \times 10^{13}} - 0,095$ [Kaplan dan kawan-kawan, 2020].

di mana a dan b adalah parameter yang diestimasi secara empiris. Meskipun sederhana, fungsi ini telah berhasil menginterpretasikan kemampuan penskalaan model bahasa dan sistem penerjemahan mesin dalam hal ukuran model (dilambangkan dengan N) dan ukuran set data pelatihan (dilambangkan dengan D).

[Gordon et al., 2021; Hestness et al., 2017]. Misalnya, Kaplan et al. [2020] menemukan bahwa kinerja model bahasa mereka meningkat sebagai hukum pangkat N atau D setelah awal

periode transien, dan menyatakan hubungan ini menggunakan $L(N) =$ dan $L(D) = \frac{N}{8,8 \times 1013}$

Perbaikan pada hukum skala ini adalah dengan menambahkan istilah kesalahan yang tidak dapat dikurangi pada hukum pangkat. bentuk $L(x)$ kemudian diberikan oleh

$$\text{Persamaan } L(x) = axb + \ddot{y}\ddot{y} \quad (2.37)$$

di mana \hat{y} adalah kesalahan tak tereduksi yang memperhitungkan kesalahan karena variabel yang tidak diketahui, yaitu hadir bahkan ketika $x \neq \hat{y}$. Persamaan (2.37) adalah salah satu bentuk yang paling banyak digunakan untuk merancang skala hukum LLM. Misalnya, [Rosenfeld et al. \[2020\]](#) mengembangkan hukum skala yang melibatkan keduanya penskalaan model dan penskalaan kumpulan data, seperti ini

$$L(N, D) = aNb + cDd + \ddots \quad (2.38)$$

Contoh dari rumusan tersebut adalah hukum penskalaan Chinchilla. Hukum ini menyatakan bahwa kehilangan uji per token adalah jumlah fungsi proporsi terbalik dari N dan D , dengan tambahan yang tidak dapat direduksi istilah kesalahan. Hoffmann et al. [2022] mengungkapkan hukum skala ini sebagai

$$L(N, D) = \frac{406.4}{\text{Nomor } 0.34} + \frac{410.7}{D0.28} + 1,69 \quad . . .$$

kesalahan yang tidak dapat diperbaiki

skala model penskalaan kumpulan data

(2.39)

Semua hukum skala yang disebutkan di atas didasarkan pada fungsi monotonik. Jadi mereka tidak dapat mencakup fungsi dengan titik belok, seperti kurva turunan ganda. Sebagai tanggapan, para peneliti telah mengeksplorasi fungsi yang lebih canggih agar sesuai dengan kurva pembelajaran. Contoh fungsi tersebut dapat

dapat ditemukan dalam karya [Alabdulmohsin et al. \[2022\]](#) dan [Caballero et al. \[2023\]](#).

Pentingnya hukum penskalaan terletak pada penyediaan arahan untuk penelitian LLM: jika kita masih berada di wilayah kurva hukum pangkat, menggunakan lebih banyak sumber daya untuk melatih model yang lebih besar adalah arah yang sangat menjanjikan. Sementara hasil ini "memaksa" kelompok penelitian dan perusahaan besar untuk berinvestasi lebih banyak dalam sumber daya komputasi untuk melatih model yang lebih besar, yang sangat mahal, hukum penskalaan terus-menerus mendorong batas-batas AI lebih jauh. Di sisi lain, memahami hukum penskalaan membantu peneliti membuat keputusan dalam melatih LLM. Misalnya, mengingat sumber daya komputasi yang ada, kinerja LLM dapat diprediksi.

Satu catatan terakhir tentang hukum penskalaan di bagian ini. Untuk LLM, kehilangan pengujian yang lebih rendah tidak selalu berarti kinerja yang lebih baik pada semua tugas hilir. Untuk mengadaptasi LLM, ada beberapa langkah seperti penyempurnaan dan prompting yang dapat memengaruhi hasil akhir. Oleh karena itu, hukum penskalaan untuk tugas hilir yang berbeda mungkin berbeda dalam praktiknya.

2.3 Pemodelan Urutan Panjang

Kita telah melihat bahwa, dalam pelatihan skala besar, model bahasa yang lebih besar dapat dikembangkan dengan menggunakan lebih banyak data dan sumber daya komputasi. Namun, peningkatan skala juga dapat terjadi dalam arah lain. Misalnya, dalam banyak aplikasi, LLM diadaptasi untuk memproses urutan yang sangat panjang. Contoh yang menarik adalah bahwa kami melakukan pra-pelatihan LLM pada teks-teks ekstensif dengan panjang normal dan kemudian menerapkannya untuk menangani token yang sangat panjang, jauh melampaui panjang yang ditemukan dalam pra-pelatihan. Di sini kita menggunakan $\Pr(y|x)$ untuk menunjukkan probabilitas pembuatan teks, di mana x adalah konteks dan y adalah teks yang dihasilkan. Secara umum ada tiga jenis masalah pemodelan sekuens panjang.

- Pembuatan teks berdasarkan konteks yang panjang (misalnya, x adalah rangkaian yang panjang). Misalnya, kami membuat ringkasan singkat untuk teks yang sangat panjang.
- Pembuatan teks panjang (misalnya, y adalah urutan yang panjang). Misalnya, kita membuat cerita panjang berdasarkan beberapa kata kunci.
- Pembuatan teks panjang berdasarkan konteks panjang (misalnya, x dan y merupakan urutan panjang). misalnya, kita menerjemahkan dokumen panjang dari bahasa Mandarin ke bahasa Inggris.

Baru-baru ini, para peneliti NLP lebih tertarik untuk menerapkan dan mengevaluasi LLM pada tugas-tugas yang melibatkan teks masukan yang sangat panjang. Bayangkan sebuah LLM, yang membaca berkas sumber C++ yang berisi puluhan ribu baris, dan menguraikan fungsionalitas program yang sesuai dengan berkas sumber tersebut. Model-model seperti itu, yang mampu menangani konteks tekstual yang luas, terkadang disebut LLM konteks panjang. Di bagian ini, kita akan membatasi diri pada LLM konteks panjang, tetapi metode yang dibahas di sini dapat diterapkan pada masalah-masalah lain.

Bagi Transformers, menangani urutan yang panjang membutuhkan komputasi yang mahal, karena biaya komputasi untuk perhatian diri sendiri tumbuh secara kuadratik seiring dengan panjang urutan. Hal ini membuat pelatihan dan penerapan model tersebut untuk input yang sangat panjang menjadi tidak layak. Dua penelitian telah mencoba mengadaptasi Transformers ke pemodelan bahasa konteks panjang.

- Yang pertama mengeksplorasi metode pelatihan yang efisien dan arsitektur model untuk mempelajari perhatian diri sendiri model dari data urutan panjang.

- Yang lain mengadaptasi LLM yang telah dilatih sebelumnya untuk menangani urutan panjang dengan penyetelan halus yang sederhana atau tanpa penyetelan halus.
- upaya.

Di sini, kami akan membahas yang pertama secara singkat karena dapat ditemukan dalam diskusi umum tentang arsitektur Transformer yang efisien [Tay et al., 2020; Xiao dan Zhu, 2023]. Kami akan fokus pada yang terakhir, dengan menyoroti metode populer dalam LLM terkini. Kami juga akan membahas kekuatan dan keterbatasan model urutan panjang ini.

2.3.1 Optimasi dari Perspektif HPC

Kami memulai diskusi dengan mempertimbangkan berbagai peningkatan pada model Transformer standar dari perspektif komputasi berkinerja tinggi. Sebagian besar peningkatan ini, meskipun tidak dirancang khusus untuk LLM, telah diterapkan secara luas di berbagai model pembelajaran mendalam [Kim et al., 2023]. Pendekatan yang umum digunakan adalah mengadopsi implementasi Transformer dengan presisi rendah. Misalnya, kita dapat menggunakan tipe data titik tetap 8-bit atau 16-bit untuk operasi aritmatika, alih-alih tipe data titik mengambang 32-bit atau 64-bit. Penggunaan tipe data dengan presisi rendah ini dapat meningkatkan efisiensi dan throughput memori, sehingga urutan yang lebih panjang dapat diproses dengan lebih mudah. Pendekatan alternatif adalah meningkatkan Transformers dengan menggunakan teknik yang mendukung perangkat keras. Misalnya, pada GPU modern, efisiensi Transformers dapat ditingkatkan dengan menggunakan implementasi fungsi self-attention yang mendukung IO [Dao et al., 2022; Kwon et al., 2023].

Cara lain untuk menangani sekuen panjang adalah melalui paralelisme sekuen [Li et al., 2023b; Korthikanti et al., 2023]. Secara khusus, pertimbangkan masalah umum menghadiri kueri q_i di posisi i ke kunci K dan nilai V . Kita dapat membagi K dengan baris dan memperoleh satu set sub-matriks $\{K[1], \dots, K[n]\}$, yang masing-masing sesuai dengan segmen sekuen. Demikian pula, kita dapat memperoleh sub-matriks V , dilambangkan dengan $\{V[1], \dots, V[n]\}$. Kemudian, kita menetapkan setiap pasangan $K[u]$ dan $V[u]$ ke node komputasi (misalnya, GPU dari kluster GPU). Node yang ditetapkan dapat berjalan secara paralel, dengan demikian memparalelkan operasi perhatian.

Ingin kembali bahwa output dari model perhatian diri dapat dituliskan sebagai

$$\text{Att}^{qkv}(q_i, K, V) = \sum_{j=0}^{\dim{V}} \frac{q_i \cdot k_j}{\sqrt{d}} + \text{Mask}(i, j) \quad (2.40)$$

di mana $\tilde{y}_{i,j}$ adalah bobot perhatian antara posisi i dan j . Dalam Transformer, $\tilde{y}_{i,j}$ diperoleh dengan skala ulang dari produk titik antara q_i dan k_j . Skor perhatian antara q_i dan k_j . menormalisasi versi Kita memiliki

$$\tilde{y}_{i,j} = \frac{q_i \cdot k_j}{\sqrt{d}} + \text{Mask}(i, j) \quad (2.41)$$

di mana $\text{Mask}(i, j)$ adalah variabel masking untuk (i, j) . Kemudian, kita mendefinisikan bobot perhatian $\tilde{y}_{i,j}$ untuk

menjadi

$$\begin{aligned} \tilde{y}_{i,j} &= \text{Softmax}(\tilde{y}_{i,j}) \\ &= \frac{\exp(\tilde{y}_{i,j})}{\sum_j \exp(\tilde{y}_{i,j})} \end{aligned} \quad (2.42)$$

Pada setiap simpul komputasi, kita perlu menerapkan persamaan ini. Dengan kunci dan nilai yang ditetapkan pada simpul ini, menghitung pembilang sisi kanan Persamaan (2.42) (yaitu, $\exp(\tilde{y}_i, j)$) mudah, karena semua informasi yang diperlukan disimpan pada simpul tersebut. Namun, menghitung penyebut sisi kanan Persamaan (2.42) melibatkan penjumlahan $\exp(\tilde{y}_i, \tilde{y}_j)$ atas semua j yang memerlukan s , yang mana pemindahan data ke dan dari simpul lain. Sebagai ilustrasi, misalkan v_j dan k_j ditempatkan pada simpul u . Kita dapat menulis ulang Persamaan (2.42) sebagai

$$\begin{aligned} & \text{aku}, j \\ & = \frac{\exp(\tilde{y}_i, j)}{\exp(\tilde{y}_i, \tilde{y}_1) + \dots + \exp(\tilde{y}_i, \tilde{y}_u) + \dots + \exp(\tilde{y}_i, \tilde{y}_n)} \quad (2.43) \\ & \quad \text{simpul } u \\ & \quad \text{--- ---} \\ & \quad \text{simpul 1} \qquad \text{simpul } u \qquad \text{simpul } n \end{aligned}$$

di mana notasi k_j \tilde{y}_j $K[u]$ menyatakan bahwa k_j \tilde{y}_j adalah vektor baris implementasi $K[u]$. Secara sederhana pertama-tama kita melakukan penjumlahan $\{k_j \tilde{y}_j K[u] \exp(\tilde{y}_i, \tilde{y}_j)\}$ secara terpisah pada node yang sesuai. Kemudian, kita mengumpulkan hasil penjumlahan ini dari node yang berbeda untuk menggabungkannya menjadi hasil akhir. Hal ini sesuai dengan operasi kolektif dalam konteks pemrosesan paralel. Ada banyak implementasi efisien dari operasi semacam itu, seperti algoritma all-reduce. Oleh karena itu, jumlah semua nilai $\exp(\tilde{y}_i, j)$ dapat dihitung menggunakan rutin yang dioptimalkan dalam perangkat komunikasi kolektif.

Mengingat bobot perhatian $\{\tilde{y}_i, j\}$, kami kemudian menghitung hasil perhatian menggunakan Persamaan (2.40). Masalahnya dapat diungkapkan kembali sebagai

$$\begin{aligned} & \text{Attqkv (qi , K, V)} \\ & = \frac{\tilde{y}_i, \tilde{y}_j v_j}{v_j \tilde{y}_j V[1] + \dots + v_j \tilde{y}_j V[u] + \dots + v_j \tilde{y}_j V[nu]} \quad (2.44) \\ & \quad \text{jika } v_j \neq 0 \\ & \quad \text{simpul 1} \qquad \text{simpul } u \qquad \text{simpul } n \end{aligned}$$

Seperi Persamaan (2.43), Persamaan (2.44) dapat diimplementasikan sebagai program penjumlahan dalam pemrosesan paralel. Pertama, lakukan penjumlahan nilai tertimbang pada node yang berbeda secara bersamaan. Kemudian, kami kumpulkan hasil dari node ini melalui operasi kolektif.

Perlu dicatat bahwa, meskipun bagian ini terutama berfokus pada pemodelan sekvens panjang, sebagian besar motivasi untuk paralelisme sekvens berasal dari metode pelatihan terdistribusi dari jaringan dalam, seperti yang dibahas dalam Bagian 2.2.3. Akibatnya, penerapan metode ini dapat didasarkan pada pustaka pemrosesan paralel yang sama.

2.3.2 Arsitektur yang Efisien

Salah satu kesulitan dalam menerapkan Transformer pada urutan yang panjang adalah bahwa self-attention memiliki kompleksitas waktu kuadratik sehubungan dengan panjang urutan. Selain itu, cache kunci-nilai (atau cache KV singkatnya) dipertahankan selama inferensi, dan ukurannya meningkat seiring dengan semakin banyak token yang diproses. Meskipun cache KV tumbuh secara linear seiring dengan panjang urutan, untuk urutan input yang sangat panjang, jejak memori menjadi signifikan dan bahkan tidak layak untuk menggunakan LLM. Oleh karena itu, arsitektur model LLM konteks panjang umumnya menjauh dari model standar

Transformer, beralih ke pengembangan varian dan alternatif yang lebih efisien.

Salah satu pendekatannya adalah dengan menggunakan sparse attention alih-alih standard self-attention. Rangkaian model ini didasarkan pada gagasan bahwa hanya sejumlah kecil token yang dianggap penting saat memperhatikan token tertentu, sehingga sebagian besar bobot perhatian antara token mendekati nol. Sebagai konsekuensinya, kita dapat memangkas sebagian besar bobot perhatian dan merepresentasikan model perhatian dalam bentuk yang ringkas. Sebagai ilustrasi, pertimbangkan model perhatian diri

$$\text{Attqkv}(Q, K, V) = \tilde{y}(Q, K)V \quad (2.45)$$

di mana matriks bobot perhatian $\tilde{y}(Q, K)$ $\in \mathbb{R}^{m \times m}$ diperoleh dengan

$$\begin{aligned} \tilde{y}(Q, K) &= \text{Softmax}(QKT) + \text{Masker} \\ &= \begin{bmatrix} \text{dari } 0,0 & & & & & \\ \vdots & \ddots & & & & \\ \text{dari } 1,0 & \tilde{y}_{1,1} & & & & \\ \vdots & \vdots & \ddots & & & \\ \text{dari } 2,0 & \tilde{y}_{2,1} & \tilde{y}_{2,2} & \cdots & & \\ \vdots & \vdots & \vdots & \ddots & & \\ & \tilde{y}_{m,1}, \tilde{y}_{m,1}, \tilde{y}_{m,1}, \dots, \tilde{y}_{m,1} & & & & \end{bmatrix} \quad (2.46) \end{aligned}$$

Setiap vektor baris $\tilde{y}_{i,0} \dots \tilde{y}_{i,i} \dots 0$ berkorespondensi dengan distribusi perhatian token ke-*i* terhadap setiap token dalam urutan. Karena model bahasa memprediksi token berikutnya hanya berdasarkan konteks kirinya, kita biasanya menulis output model perhatian pada posisi *i* sebagai

$$\begin{aligned} \text{Attqkv}(q_i, K_{\tilde{y}_i}, V_{\tilde{y}_i}) &= \tilde{y}_{i,0} \dots \tilde{y}_{i,i} \\ &= \begin{bmatrix} \vdots & \vdots \\ \text{kita} & \end{bmatrix} \\ &= \begin{bmatrix} \text{aku, } jv_j \\ \vdots \\ \text{kita} \end{bmatrix} \quad (2.47) \end{aligned}$$

dimana $K_{\tilde{y}_i} = \begin{bmatrix} \vdots \\ \text{aku} \end{bmatrix}$ dan $V_{\tilde{y}_i} = \begin{bmatrix} \vdots \\ \text{kita} \end{bmatrix}$ adalah kunci dan nilai hingga posisi *i*.

Dalam versi asli dari perhatian-diri, $\tilde{y}_{i,0} \dots \tilde{y}_{i,i}$ diasumsikan padat, yaitu, sebagian besar nilainya bukan nol. Dalam perhatian yang jarang, beberapa entri dari $\tilde{y}_{i,0} \dots \tilde{y}_{i,i}$ dianggap bukan nol, dan entri yang tersisa diabaikan begitu saja dalam perhitungan. Misalkan $G \subseteq \{0, \dots, i\}$ adalah himpunan indeks entri bukan nol. Untuk model bahasa, keluaran dari model perhatian yang jarang pada posisi *i* diberikan oleh

$$\text{Attsparse}(q_i, K_{\tilde{y}_i}, V_{\tilde{y}_i}) = \sum_{j \in G} \frac{\text{aku, } jv_j}{\text{jumlah } j \in G} \quad (2.48)$$

Di sini $\frac{\text{aku, } jv_j}{\text{jumlah } j \in G}$ dinormalisasi atas G . Oleh karena itu, nilainya berbeda dari nilai aslinya (perhatikan > bobot \tilde{y} pada kenyataannya kita memiliki $\tilde{y}_{i,j}$). Kelangkaan model ditentukan oleh seberapa besar G .

Model perhatian jarang berbeda dalam cara kita mendefinisikan G . Salah satu pendekatan sederhana adalah mendefinisikan G berdasarkan

pada pola yang dirancang secara heuristik. Misalnya, pola yang banyak digunakan melibatkan G yang menutupi jendela token yang terletak di dekat posisi i [Parmar et al., 2018].

Meskipun sparse attention mengurangi komputasi melalui penggunaan operasi sparse, model tersebut masih memiliki keterbatasan yang signifikan karena kita harus menyimpan seluruh cache KV (yaitu, K_{ij} dan V_{ij}) selama inferensi. Jika urutannya sangat panjang, penyimpanan cache ini akan menjadi sangat intensif memori. Untuk mengatasi hal ini, kita dapat mempertimbangkan bentuk model perhatian yang berbeda di mana cache KV tidak disimpan secara eksplisit. Perhatian linier adalah salah satu pendekatan tersebut [Katharopoulos et al., 2020]. Ia menggunakan fungsi kernel $\hat{y}(\cdot)$ untuk memproyeksikan setiap kueri dan kunci ke titik $q = \hat{y}(q_i)$ dan $k = \hat{y}(k_i)$, masing-masing. Dengan menghilangkan fungsi Softmax dalam transformasi tersebut¹⁰ bentuk dari model perhatian yang dihasilkan diberikan oleh

$$\text{Att}_{\text{kv}}(q_i, K_{ij}, V_{ij}) \hat{y} \text{ Att}_{\text{linear}}(q_i, V_{ij})$$

$$= \frac{\sum_{k=1}^n q_i \hat{y}(q_i) K_{kj} \hat{y}(k_i)}{\sum_{k=1}^n \hat{y}(k_i)^2} \quad (2.49)$$

di mana \hat{y}_i dan \hat{y}_j adalah variabel yang dihitung dalam bentuk berulang

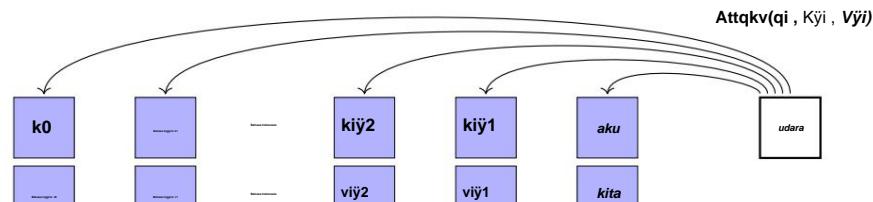
$$\hat{y}_i = \hat{y}_i \hat{y}_1 + k_i v_i \quad (2.50)$$

$$\hat{y}_j = \hat{y}_j \hat{y}_1 + k_j v_j \quad (2.51)$$

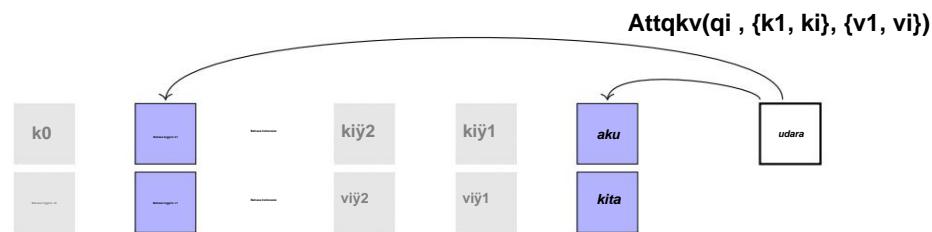
\hat{y}_i dan \hat{y}_j dapat dilihat sebagai representasi riwayat hingga posisi i . Manfaat model ini adalah kita tidak perlu menyimpan semua kueri dan nilai sebelumnya. Sebaliknya, hanya representasi terbaru \hat{y}_i dan \hat{y}_j yang digunakan. Jadi, biaya komputasi setiap langkah adalah konstan, dan model dapat dengan mudah diperluas untuk menangani urutan yang panjang.

Faktanya, pendekatan sekuensial terhadap pemodelan urutan panjang ini muncul secara alami ketika kita mengadopsi sudut pandang model berulang. Model tersebut membaca satu token (atau sejumlah kecil token) pada satu waktu, memperbarui status berulang menggunakan masukan ini, lalu membuangnya sebelum token berikutnya tiba. Keluaran pada setiap langkah dihasilkan hanya berdasarkan status berulang, bukan pada semua status sebelumnya. Jejak memori ditentukan oleh status berulang yang memiliki ukuran tetap. Model berulang dapat digunakan dalam skenario pembelajaran waktu nyata di mana data tiba dalam aliran dan prediksi dapat dibuat pada setiap langkah waktu. Dalam NLP, menerapkan model berulang pada pemodelan bahasa adalah salah satu upaya awal yang berhasil untuk mempelajari representasi urutan. Meskipun Transformer telah digunakan sebagai arsitektur dasar dalam LLM, model berulang masih merupakan model yang kuat, terutama untuk mengembangkan LLM yang efisien. Baru-baru ini, model berulang telah memulai kebangkitannya dalam pemodelan bahasa dan telah dipertimbangkan kembali sebagai alternatif yang menjanjikan untuk Transformer [Gu dan Dao, 2023]. Gambar 2.5 menunjukkan perbandingan model yang dibahas dalam subbagian ini.

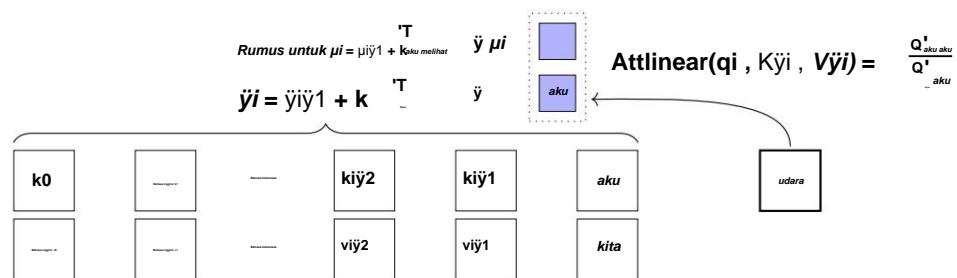
¹⁰Di ruang baru setelah transformasi ini, normalisasi Softmax dapat diubah menjadi penskalaan sederhana normalisasi.



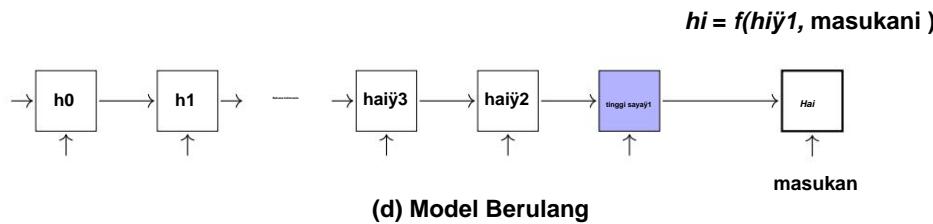
(a) Standar Perhatian Diri



(b) Kurangnya Perhatian



(c) Perhatian Linear



Gambar 2.5: Ilustrasi model perhatian diri, perhatian jarang, perhatian linier dan berulang. Kotak biru = di-cache

negara untuk menghasilkan keluaran pada posisi i . $f(\cdot)$ = sel berulang.

2.3.3 Cache dan Memori

LLM yang didasarkan pada arsitektur Transformer standar adalah model global. Inferensi untuk model ini melibatkan penyimpanan seluruh konteks kiri untuk membuat prediksi untuk token masa depan. Ini membutuhkan cache KV di mana representasi (yaitu, kunci dan nilai) dari semua yang dihasilkan sebelumnya

token disimpan, dan biaya penyimpanan sementara bertambah seiring dengan berlanjutnya inferensi. Di atas, kami telah membahas metode untuk mengoptimalkan penyimpanan sementara ini melalui pendekatan perhatian yang efisien, seperti perhatian yang jarang dan perhatian linier. Ide lain, yang mungkin tumpang tindih dengan pembahasan sebelumnya, adalah mengodekan konteks secara eksplisit melalui model memori tambahan.

2.3.3.1 Cache KV Berukuran Tetap

Pendekatan yang langsung adalah dengan merepresentasikan kunci dan nilai menggunakan model memori berukuran tetap. Misalkan kita memiliki memori Mem yang menyimpan informasi kontekstual. Kita dapat menulis operasi perhatian pada posisi i dalam bentuk umum

$$\text{Att}(q_i, \text{Mem}) = \text{Att}_{\text{KV}}(q_i, K_{ij}, V_{ij}) \quad (2.52)$$

Dalam model ini, Mem hanyalah cache KV, yaitu, $\text{Mem} = (K_{ij}, V_{ij})$. Jadi, ukuran Mem ditentukan oleh i . Jika kita mendefinisikan Mem sebagai variabel berukuran tetap, maka biaya pelaksanaan $\text{Att}(q_i, \text{Mem})$ akan tetap. Ada beberapa cara alternatif untuk mendesain Mem.

- Salah satu metode paling sederhana adalah mempertimbangkan jendela berukuran tetap dari kunci dan nilai sebelumnya. Oleh karena itu Mem diberikan oleh

$$\text{Mem} = (K[i:j], V[i:j]) \quad (2.53)$$

di mana nc menunjukkan ukuran jendela. Notasi $K[i:j]$ dan $V[i:j]$ menunjukkan kunci dan nilai pada posisi dari i hingga j .
¹¹ Model ini dapat dilihat sebagai jenis model perhatian lokal.

- Mem juga dapat didefinisikan sebagai sepasang vektor ringkas, yang mengarah ke representasi sejarah yang lebih ringkas. Cara sederhana untuk meringkas kunci dan nilai sebelumnya adalah dengan menggunakan rata-rata bergeraknya. Misalkan, Mem dapat didefinisikan sebagai rata-rata bergerak tanpa bobot dari kunci dan nilai nc sebelumnya.

$$\text{Mem} = \frac{\sum_{j=i-nc+1}^i k_j}{nc} \quad \frac{\sum_{j=i-nc+1}^i v_j}{nc} \quad (2.54)$$

Atau, kita dapat menggunakan versi tertimbang dari rata-rata bergerak

$$\text{Mem} = \frac{\sum_{j=1}^{i-nc+1} w_j k_j}{\sum_{j=1}^{i-nc+1} w_j} \quad \frac{\sum_{j=1}^{i-nc+1} w_j v_j}{\sum_{j=1}^{i-nc+1} w_j} \quad (2.55)$$

Di sini $\{y_1, \dots, y_{nc}\}$ adalah koefisien, yang dapat dipelajari sebagai parameter model atau ditentukan melalui heuristik. Misalkan, mereka dapat diatur ke koefisien yang meningkat (yaitu, $y_1 < y_2 < \dots < y_{nc}$) untuk memberikan bobot yang lebih besar pada posisi yang lebih dekat ke

11 Lebih formalnya, kita menulis $K[i:j] = \begin{bmatrix} k_{i-nc+1} \\ \vdots \\ k_i \end{bmatrix}$ dan $V[i:j] = \begin{bmatrix} v_{i-nc+1} \\ \vdots \\ v_i \end{bmatrix}$ derkadang-kadang kita menandainya

$K[i:j]$ oleh $\{k_{i-nc+1}, \dots, k_i\}$ dan $V[i:j]$ oleh $\{v_{i-nc+1}, \dots, v_i\}$ untuk kesederhanaan notasi.

i. Kita dapat memperluas rata-rata bergerak untuk mencakup semua posisi hingga i . Hal ini mengarah pada rata-rata kumulatif kunci dan nilai, yang diberikan dalam bentuk

$$\text{Mem} = \frac{\sum_{j=0}^{k_j(i)} v_j}{j+1} - \frac{\sum_{j=0}^{v_j(i)} k_j}{j+1} \quad (2.56)$$

Secara umum, rata-rata kumulatif dapat ditulis menggunakan rumus rekursif

$$\text{Mem}_i = \frac{(k_i, v_i) + \text{say}_i \cdot \text{Mem}_{i-1}}{\text{say}_i + 1} \quad (2.57)$$

di mana Mem_i dan Mem_{i-1} masing-masing menunjukkan rata-rata kumulatif dari posisi saat ini dan sebelumnya. Keuntungan dari model ini adalah kita hanya perlu menyimpan satu pasangan kunci-nilai selama inferensi, daripada menyimpan semua pasangan kunci-nilai. Perhatikan bahwa model memori di atas terkait dengan model berulang, dan teknik yang lebih maju telah digunakan untuk mengembangkan alternatif mekanisme self-attention dalam Transformers [Ma et al., 2023].

- Memori Mem juga dapat berupa jaringan saraf. Pada setiap langkah, ia mengambil keluaran memori sebelumnya dan status model saat ini sebagai masukan, dan menghasilkan keluaran memori yang baru. Jaringan saraf ini dapat diformulasikan sebagai fungsi

$$\text{Mem} = \text{Perbarui}(\text{Skv}, \text{Mempre}) \quad (2.58)$$

Di sini Mem dan Mempre masing-masing mewakili keluaran memori pada langkah saat ini dan langkah sebelumnya. Skv adalah sekumpulan pasangan kunci-nilai, yang mewakili status model terkini. Rumusan ini bersifat umum dan memungkinkan kita mengembangkan berbagai model memori dengan memilih konfigurasi $\text{Update}(\cdot)$ dan Skv yang berbeda. Misalnya, jika Skv hanya berisi pasangan kunci-nilai terkini (k_i, v_i) dan $\text{Update}(\cdot)$ didefinisikan sebagai sel berulang, maka Persamaan (2.58) dapat dinyatakan sebagai model mirip RNN.

$$\text{Mem} = f((k_i, v_i), \text{Mempre}) \quad (2.59)$$

di mana $f(\cdot)$ adalah sel berulang. Pengulangan juga dapat diterapkan pada pemodelan tingkat segmen untuk pertimbangan efisiensi. Pendekatan sederhana adalah kita dapat membagi urutan menjadi segmen, dan memperlakukan Skv sebagai segmen. Menerapkan model berulang ke $\text{Update}(\cdot)$ akan menghasilkan model memori yang beroperasi pada segmen. Contoh khusus adalah kita mendefinisikan $\text{Update}(\cdot)$ sebagai fungsi FIFO yang menambahkan Skv ke dalam memori dan menghapus segmen nilai kunci tertua dari memori, yang diberikan oleh

$$\text{Mem} = \text{FIFO}(\text{Skv}, \text{Mempre}) \quad (2.60)$$

Pertimbangkan memori yang mencakup dua segmen, satu untuk segmen saat ini, dan satu untuk segmen sebelumnya. Dalam operasi perhatian, setiap posisi dapat mengakses pasangan kunci-nilai riwayat dalam dua segmen berurutan terdekat. Ini pada dasarnya mendefinisikan memori lokal, tetapi memori dan variannya telah banyak digunakan dalam model rekursif tingkat segmen [Dai et al., 2019; Hutchins et al., 2022; Bulatov et al., 2022].

- Model memori di atas dapat diperluas untuk melibatkan beberapa memori. Contoh

Pendekatan ini menggunakan Transformator Kompresif [Rae et al., 2019]. Ia menggunakan dua memori berukuran tetap yang berbeda: satu untuk pemodelan konteks lokal (dilambangkan dengan Mem), dan yang lainnya untuk pemodelan dan kompresi riwayat jangka panjang (dilambangkan dengan CMem). Cache KV dalam model ini adalah kombinasi dari Mem dan CMem. Fungsi perhatian dapat dituliskan sebagai

$$\text{Attcom}(\mathbf{q}_i, \text{Mem}, \text{CMem}) = \text{Attqkv}(\mathbf{q}_i, [\text{Mem}, \text{CMem}]) \quad (2.61)$$

di mana $[\text{Mem}, \text{CMem}]$ adalah gabungan memori dari Mem dan CMem. Seperti model tingkat segmen lainnya, model Transformer kompresif beroperasi pada segmen urutan.

Setiap segmen adalah urutan ns token yang berurutan, dan kami menyatakan S sebagai pasangan kunci-nilai yang sesuai dengan token segmen ke- k . Ketika segmen baru tiba, Mem diperbarui dalam mode FIFO: kami menambahkan pasangan kunci-nilai nc di S dan memasukkan ns pasangan kunci-nilai nv ke Mem, dan kemudian tertua dari Mem, yang diberikan oleh

$$\text{Mem} = \text{FIFO}(S_{kv}, \text{Mem}_{pre}) \quad (2.62)$$

Pasangan kunci-nilai yang dikeluarkan kemudian digunakan untuk memperbarui CMem memori kompresif.

Pasangan kunci-nilai ns ini dikompresi menjadi pasangan kunci-nilai nv melalui jaringan kompresi.

CMem adalah FIFO yang menambahkan pasangan kunci-nilai nv ke ekor antrian, dan membuang ns pasangan kunci-nilai pertama dari antrian. Hal ini diberikan oleh

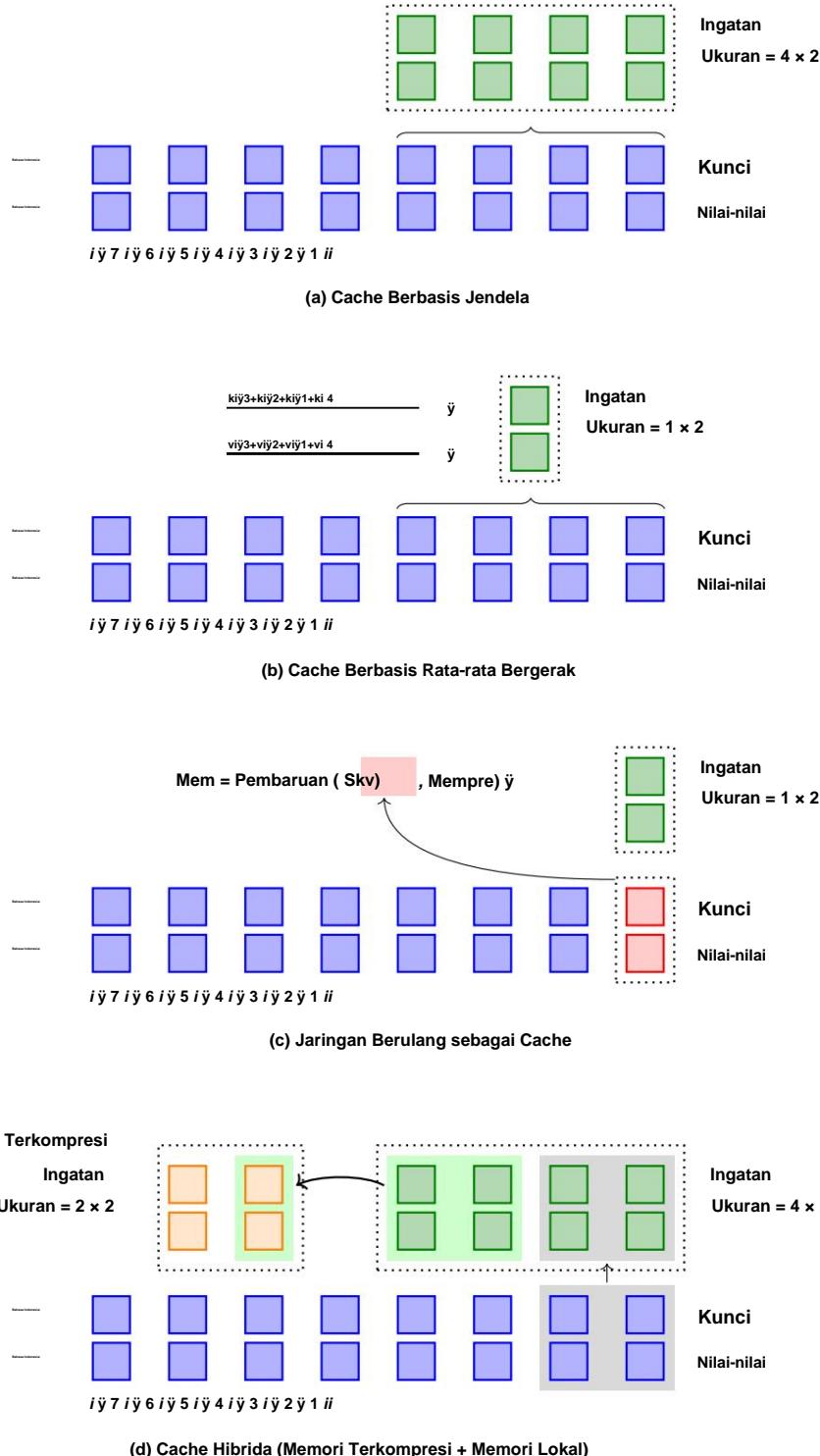
$$\text{CMem} = \text{FIFO}(C_{kv}, \text{CMem}_{pre}) \quad (2.63)$$

di mana C_{kv}^k mewakili himpunan pasangan kunci-nilai terkompresi. Tersirat dalam model Transformer kompresif adalah bahwa konteks lokal harus direpresentasikan secara eksplisit dengan kehilangan informasi minimal, sementara konteks jarak jauh dapat lebih terkompresi.

- Kita telah melihat bahwa konteks global dan lokal sama-sama berguna dan dapat dimodelkan menggunakan model perhatian. Pandangan ini memotivasi perluasan model perhatian untuk menggabungkan memori lokal dan jangka panjang [Ainslie et al., 2020 ; Zaheer et al., 2020; Gupta dan Berant, 2020]. Pendekatan yang sederhana tetapi banyak digunakan adalah melibatkan beberapa token pertama dari urutan tersebut dalam perhatian, yang berfungsi sebagai token global. Pendekatan ini biasanya diterapkan bersama dengan model perhatian jarang lainnya. Keuntungan dari penggabungan token global dari urutan tersebut adalah membantu memperlancar distribusi keluaran fungsi Softmax yang digunakan dalam komputasi bobot perhatian, dan dengan demikian menstabilkan kinerja model ketika ukuran konteks sangat besar [Xiao et al., 2024]. Namun, satu kelemahannya adalah bahwa menggunakan memori global berukuran tetap dapat mengakibatkan hilangnya informasi. Ketika berhadapan dengan urutan yang panjang, kita perlu memperbesar cache KV untuk representasi konteks yang memadai, tetapi hal ini pada gilirannya meningkatkan biaya komputasi.

Gambar 2.6 menunjukkan ilustrasi pendekatan di atas. Perhatikan bahwa, meskipun kami berfokus pada pengoptimalan cache KV di sini, masalah ini terkait erat dengan masalah yang dibahas di bagian sebelumnya. Semua metode yang telah kami sebutkan sejauh ini secara luas dapat dikategorikan sebagai pendekatan perhatian efisien, yang digunakan secara luas dalam berbagai varian Transformer.

2.3 Pemodelan Urutan Panjang



Gbr. 2.6: Ilustrasi cache KV berukuran tetap dalam LLM. Kotak biru mewakili kunci dan nilai yang dihasilkan selama inferensi LLM, kotak hijau mewakili kunci dan nilai yang disimpan atau dikodekan dalam memori utama, dan kotak oranye mewakili kunci dan nilai yang disimpan atau dikodekan dalam memori terkompresi.

2.3.3.2 Model Berbasis Memori

Pemodelan memori yang dibahas di atas didasarkan pada pembaruan cache KV, dan model yang dihasilkan biasanya disebut sebagai memori internal. Sekarang kita pertimbangkan keluarga model lain, yang disebut memori eksternal, yang beroperasi sebagai model independen untuk mengakses konteks skala besar untuk LLM. Banyak model tersebut didasarkan pada metode berbasis memori yang telah dibahas secara luas dalam pembelajaran mesin [Bishop, 2006]. Contoh umum adalah algoritma tetangga terdekat: kita menyimpan representasi konteks dalam penyimpanan data, dan mencoba menemukan representasi tersimpan yang paling mirip untuk mencocokkan kueri yang diberikan. Representasi konteks yang diambil kemudian digunakan untuk meningkatkan perhatian terhadap kueri ini.

Di sini, kami mempertimbangkan metode k-nearest neighbours (k-NN) yang merupakan salah satu metode berbasis memori yang paling populer. Karena fokus kami adalah pemodelan bahasa di bagian ini, kami mendefinisikan sampel dalam penyimpanan data sebagai pasangan kunci-nilai yang sesuai dengan beberapa status konteks. Perhatikan bahwa "konteks" adalah konsep yang luas di sini, bukan hanya awalan sekuens dalam pembuatan teks. Seseorang mungkin, misalnya, melihat seluruh kumpulan data sebagai konteks untuk memprediksi token. Ini memungkinkan kami untuk mengambil situasi konteks terdekat dalam satu set sekuens, daripada awalan sekuens tertentu. Meskipun kami akan membatasi diri pada pemodelan konteks untuk satu sekuens, dalam subbagian ini, kami membahas kasus yang relatif lebih umum.

Misalkan kita memiliki sekumpulan kunci $\{kj\}$ dengan nilai terkait $\{vj\}$, dan misalkan kita menyimpan pasangan kunci-nilai ini dalam basis data vektor¹². Untuk setiap kueri qi , kita temukan k tetangga terdekatnya dengan mengembangkan jari-jari bola yang berpusat sebagai qi hingga berisi k titik data dalam $\{kj\}$. Ini menghasilkan sekumpulan k kunci beserta nilai terkaitnya, dilambangkan dengan *Memknn*. Seperti sebelumnya, kita menyatakan *Mem* sebagai memori lokal untuk kueri, seperti cache KV dari token tetangga. Sasaran kita adalah untuk memberikan kueri qi pada memori lokal *Mem* dan memori jangka panjang *Memknn*.

Tentu saja, ada beberapa cara untuk menggabungkan *Mem* dan *Memknn* ke dalam model perhatian.

Misalnya, kita dapat menggabungkan keduanya untuk membentuk cache KV tunggal [*Mem*, *Memknn*], dan memperhatikan qi ke [*Mem*, *Memknn*] melalui perhatian QKV standar. Atau kita dapat menggunakan *Mem* dan *Memknn* dalam langkah perhatian terpisah. Contoh pendekatan tersebut adalah model yang dikembangkan oleh Wu et al. [2021]. Ini secara linier menggabungkan dua jenis perhatian, yang diberikan oleh

$$\text{Att}(qi, \text{Mem}, \text{Memknn}) = g \circ \text{Attlokal} + (1 - g) \circ \text{Attknn} \quad (2.64)$$

$$\text{Attknn} = \text{Att}(qi, \text{Memknn}) \quad (2.65)$$

$$\text{Attlokal} = \text{Att}(qi, \text{Mem}) \quad (2.66)$$

Di sini $g \in \mathbb{R}^D$ adalah vektor koefisien, yang dapat berupa keluaran gerbang yang dipelajari.

Mengingat model memori berbasis k-NN yang dijelaskan di atas, tugas yang tersisa adalah menentukan pasangan kunci-nilai mana yang disimpan dalam penyimpanan data. Untuk tugas pemodelan bahasa standar, kami menganggap token yang terlihat sebelumnya dalam urutan sebagai konteks, sehingga kami dapat menambahkan kunci dan nilai dari semua token ini ke dalam penyimpanan data. Dalam hal ini, model perhatian berbasis k-NN yang dihasilkan pada dasarnya setara dengan model perhatian yang jarang [Gupta et al., 2021].

Sebagai alternatifnya, kita dapat memperluas konteks dari satu rangkaian ke kumpulan rangkaian. Misalnya, kita dapat mengumpulkan semua pasangan kunci-nilai di seluruh rangkaian dalam set data pelatihan dan menambahkannya ke penyimpanan data untuk memodelkan konteks yang lebih besar. Dengan demikian, LLM dapat memprediksi token berdasarkan

¹²Basis data vektor, atau penyimpanan vektor, adalah basis data yang menyediakan antarmuka pengambilan yang sangat optimal untuk menemukan vektor tersimpan yang sangat cocok dengan vektor kueri.

konteks umum. Masalah dengan pendekatan ini adalah biaya komputasinya akan besar jika banyak urutan yang terlibat. Karena urutan ini adalah bagian dari data pelatihan kita, kita dapat membangun dan mengoptimalkan indeks untuk vektor di penyimpanan data sebelum menjalankan LLM. Hasilnya, pengambilan vektor yang serupa dapat sangat efisien, seperti pada sebagian besar basis data vektor.

Faktanya, semua metode yang disebutkan di atas dapat dilihat sebagai contoh pendekatan berbasis pengambilan kembali. Alih-alih menggunakan hasil pengambilan kembali untuk meningkatkan perhatian, kita dapat menerapkan pendekatan ini di bidang lain. Salah satu aplikasi pencarian berbasis k-NN adalah pemodelan bahasa k-NN (atau k-NN LM)

[Khandelwal et al., 2020]. Idenya adalah, meskipun mencoba memperluas konteks yang digunakan dalam perhatian diri dengan menggabungkan tetangga terdekat dalam pembelajaran representasi, dalam praktiknya, serupa keadaan tersembunyi dalam Transformer sering kali sangat prediktif terhadap token serupa di posisi berikutnya.

Dalam k-NN LM, setiap item dalam datastore adalah tuple kunci-nilai (z, w) , dimana z mewakili nilai tersembunyi keadaan LLM pada suatu posisi, dan w mewakili prediksi yang sesuai. Cara umum untuk membuat penyimpanan data adalah mengumpulkan vektor keluaran tumpukan lapisan Transformer dan token berikutnya yang sesuai untuk setiap posisi setiap urutan dalam kumpulan data pelatihan. Selama inferensi, kita memiliki representasi hi yang diberikan sebuah awalan. Dengan representasi ini, pertama kita mencari datastore untuk k item data yang paling cocok $\{(z_1, w_1), \dots, (z_k, w_k)\}$. Di sini $\{w_1, \dots, w_k\}$ dianggap sebagai token referensi untuk prediksi, dan dengan demikian dapat digunakan untuk memandu prediksi token berdasarkan hi .

Salah satu cara umum untuk memanfaatkan token referensi adalah dengan mendefinisikan distribusi atas kosakata

$$\text{Prknn}(\cdot|hi) = \text{Softmaks}(\vec{y}_{d0} \dots \vec{y}_d | V |) \quad (2.67)$$

di mana dV sama dengan jarak antara hi dan zj jika wj sama dengan entri ke- v dari V . Kami menggunakan \vec{y} dan sama dengan 0 fungsi linier dengan koefisien \vec{y} yang melakukan interpolasi antara distribusi berbasis pengambilan $\text{Prknn}(\cdot|hi)$ dan distribusi keluaran LLM $\text{Prlm}(\cdot|hi)$

$$\text{Pr}(\cdot|hi) = \vec{y} \cdot \text{Prknn}(\cdot|hi) + (1 - \vec{y}) \cdot \text{Prlm}(\cdot|hi) \quad (2.68)$$

Kemudian, seperti biasa, kita dapat memilih token y berikutnya dengan memaksimalkan probabilitas $\text{Pr}(y|hi)$.

Seperti halnya sistem pencarian informasi (IR), penyimpanan data juga dapat mengelola teks dan menyediakan akses ke teks yang relevan untuk suatu kueri. Misalnya, kita dapat menyimpan kumpulan dokumen teks di mesin pencari dengan pengindeksan teks lengkap, lalu mencarinya untuk dokumen yang cocok dengan yang diberikan kueri berbasis teks. Penerapan teknik IR pada LLM menghasilkan kerangka kerja umum yang disebut pembangkitan augmented-retrieval (RAG). Kerangka kerja RAG bekerja sebagai berikut. Kami menggunakan konteks x sebagai kueri dan temukan k bagian dokumen paling relevan $\{c_1, \dots, c_k\}$ dari penyimpanan data melalui efisien Teknik IR13 . Hasil pencarian ini digabungkan dengan konteks asli melalui prompt

13Dalam aplikasi bajakan, kueri biasanya dibuat menggunakan sistem pembuatan kueri, yang dapat memperluasnya dengan variasi token dan maksud kueri.

templat $g(\cdot)$ 14, menghasilkan input tambahan untuk LLM

$$x' = g(c_1, \dots, c_k, x) \quad (2.69)$$

Kemudian, kami x' sebagai konteks dan prediksikan teks berikut menggunakan model $\Pr(y|x')$. Satu menggunakan x' keuntungan RAG adalah kami tidak perlu memodifikasi arsitektur LLM, tetapi sebagai gantinya menambah input ke LLM melalui sistem IR tambahan. Gambar 2.7 menunjukkan perbandingan penggunaan memori eksternal yang berbeda dalam LLM.

2.3.3.3 Kapasitas Memori

Model memori dalam LLM, dalam bentuk cache kunci-nilai sederhana atau penyimpanan data, secara umum dapat dilihat sebagai penyandi informasi kontekstual. Idealnya, sebelum kita mengatakan bahwa model memori mewakili seluruh konteks dalam prediksi token, kita perlu memastikan bahwa model tersebut dapat secara akurat mewakili bagian mana pun dari konteks tersebut. Cache KV standar adalah salah satu model yang sepenuhnya menyimpan semua riwayat masa lalu. Dalam hal ini, model tersebut dikatakan memiliki kapasitas yang memadai untuk menghafal konteks. Namun, dalam banyak aplikasi praktis, menghafal lengkap tidak diperlukan. Sebaliknya, tujuannya adalah untuk memungkinkan LLM mengakses informasi kontekstual yang penting. Hasilnya, model memori yang efisien dan terkompresi dikembangkan, seperti yang dijelaskan dalam bagian ini. Perhatikan bahwa, semakin panjang urutannya, semakin sulit bagi model memori berkapasitas rendah untuk menangkap informasi kontekstual yang penting. Oleh karena itu, merupakan praktik umum untuk sekadar meningkatkan kapasitas model saat memproses konteks yang panjang.

Meskipun model berkapasitas tinggi umumnya menguntungkan, model tersebut sulit dilatih dan diterapkan. Skenario yang menantang adalah token datang dalam aliran dan konteksnya terus berkembang. Mengembangkan LLM untuk tugas-tugas seperti itu sulit karena kita perlu melatih Transformer pada urutan yang sangat panjang. Salah satu cara yang mungkin untuk mengatasi kesulitan ini adalah dengan menggunakan metode non-parametrik, seperti metode berbasis pengambilan. Misalnya, seperti yang dibahas di atas, kita dapat menggunakan basis data vektor untuk menyimpan pasangan kunci-nilai yang dihasilkan sebelumnya, dan dengan demikian merepresentasikan konteks dengan model memori eksternal ini. Meskipun pendekatan ini menghindari tantangan merepresentasikan konteks yang panjang dalam Transformer, membangun dan memperbarui model memori eksternal membutuhkan komputasi yang mahal. Model-model ini lebih sering digunakan dalam masalah di mana konteks diberikan terlebih dahulu dan ditetapkan selama inferensi, dan karenanya tidak cocok untuk

Dalam kasus di mana ukuran konteks terus bertambah, penerapan model memori berukuran tetap merupakan pendekatan yang umum digunakan. Misalnya, dalam model berulang, urutan dengan panjang yang berubah-ubah dapat diringkas menjadi serangkaian status tersembunyi yang dengannya kita memiliki biaya komputasi tetap per langkah. Sementara model berulang awalnya ditemukan tidak begitu baik dalam menangani ketergantungan jarak jauh dalam pemodelan urutan pada aplikasi awal pembelajaran mendalam untuk NLP, kemajuan terkini telah menunjukkan bahwa variannya kini efektif dalam pemodelan urutan yang sangat panjang.

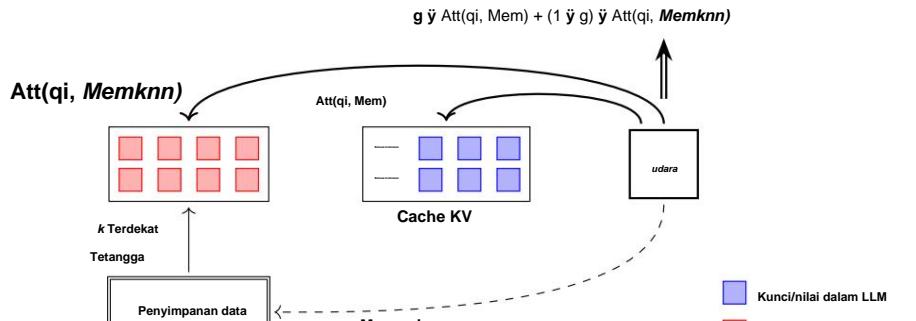
[Bulatov dkk., 2022; Hutchins dkk., 2022; Munkhdalai dkk., 2024; Ma dkk., 2024].

¹⁴Misalnya, templatnya bisa seperti ini:

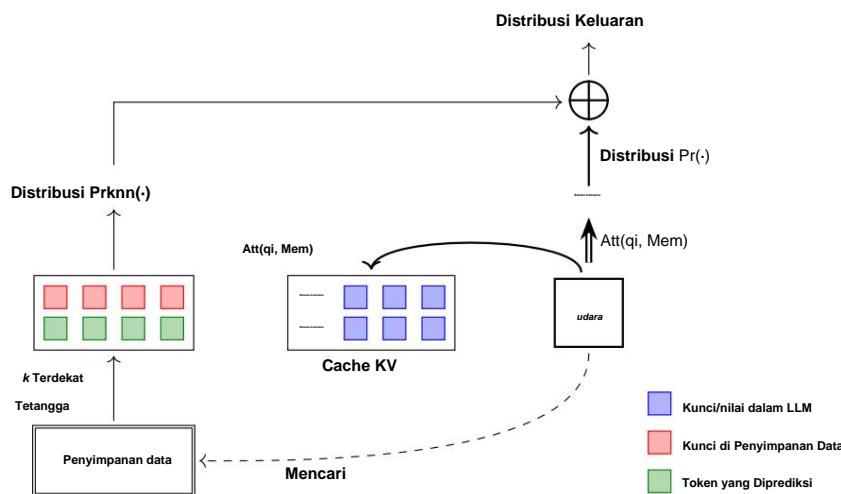
```
pesan = {"c1"} ... {"ck"} masukan:  
{"x"}  
keluaran: _____
```

2.3 Pemodelan Urutan Panjang

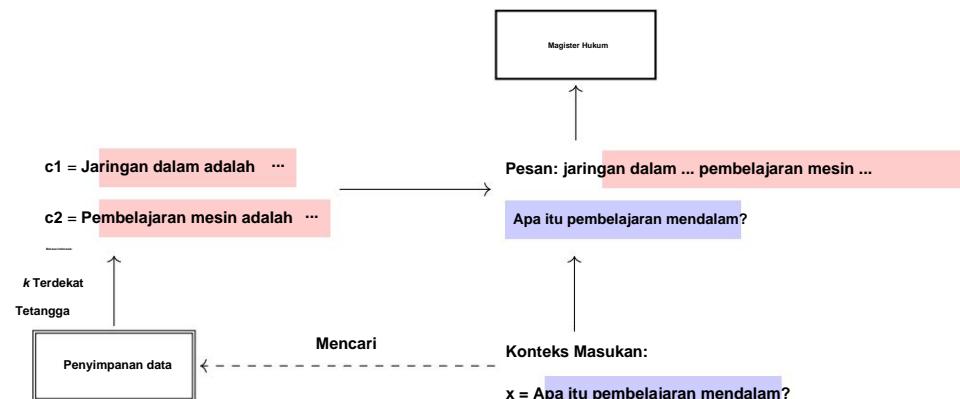
79



(a) Pencarian k-NN dengan Perhatian Tambahan



(b) Pemodelan Bahasa k-NN



(c) Generasi yang ditingkatkan dengan pengambilan kembali

Gambar 2.7: Ilustrasi memori eksternal (atau penyimpanan data) untuk pemodelan bahasa.

Tidak ada definisi umum kapasitas memori dalam LLM. Pendekatan sederhana mungkin mempertimbangkan berapa banyak penyimpanan yang digunakan untuk menyimpan informasi kontekstual. Misalnya, kapasitas memori dapat ditentukan oleh ukuran cache KV di Transformers atau basis data vektor yang digunakan dalam metode berbasis pengambilan. Konsep terkait adalah kompleksitas model. Dalam pembelajaran mesin, ada beberapa cara untuk menentukan kompleksitas model dari suatu model. Salah satu metode paling sederhana adalah dengan menghitung jumlah parameter. Namun, harus ditekankan bahwa model memori yang dibahas di sini terutama berfungsi untuk menyimpan informasi, daripada menambahkan parameter yang dapat dilatih. Oleh karena itu, model dengan kapasitas memori yang besar belum tentu lebih kompleks. Namun demikian, dalam praktiknya menentukan kapasitas model memori tidaklah mudah. Secara umum, kita perlu mengendalikan trade-off antara memaksimalkan kinerja dan mengendalikan jejak memori.

2.3.4 Berbagi antar Kepala dan Lapisan

Dalam Transformers, cache KV adalah struktur data yang dapat disesuaikan secara dinamis di sepanjang beberapa dimensi, seperti kepala, lapisan, dan panjang urutan. Misalnya, pertimbangkan LLM dengan lapisan L . Setiap lapisan memiliki kepala perhatian \tilde{y} , dan setiap kepala menghasilkan keluaran berdimensi dh . Selama inferensi, kami menyimpan kunci dan nilai hingga m token. Kompleksitas ruang mekanisme caching ini adalah $O(L \cdot \tilde{y} \cdot dh \cdot m)$. Seperti yang telah kita lihat sebelumnya, kompleksitas ini dapat dikurangi dengan melakukan caching kunci dan nilai untuk token yang lebih sedikit. Misalnya, dalam perhatian jendela geser, jendela berukuran tetap digunakan untuk melakukan caching kunci dan nilai dalam konteks lokal. Dan model ini memiliki kompleksitas ruang $O(L \cdot \tilde{y} \cdot dh \cdot mw)$, dengan mw menjadi ukuran jendela.

Selain mengurangi m , kita juga dapat mengurangi ukuran cache KV di sepanjang dimensi lain. Pendekatan yang banyak digunakan adalah dengan memungkinkan berbagi antar kepala dalam perhatian diri multi-kepala. Ingat kembali dari Bagian 2.1.1 bahwa perhatian diri multi-kepala menggunakan beberapa set kueri, kunci, dan nilai (setiap set disebut kepala), yang masing-masing menjalankan mekanisme perhatian QKV seperti biasa. Hal ini dapat dinyatakan sebagai

$$\text{Keluaran} = \text{Gabungkan}(\text{kepala}_1, \dots, \text{kepala}_{\tilde{y}}) W \text{kepala} \quad (2.70)$$

di mana $\text{head}_j \in dh$ dihitung menggunakan fungsi perhatian QKV standar

$$\text{kepalaj} = \text{Attqkv}(q, K_{\tilde{y}}, V_{\tilde{y}}) \quad (2.71)$$

Di sini, q , $K_{\tilde{y}}$, dan $V_{\tilde{y}}$ adalah kueri, kunci, dan nilai yang diproyeksikan ke fitur ke- j subruang. Jadi model ini dapat diartikan sebagai melakukan perhatian pada sekelompok subruang fitur secara paralel (lihat Gambar 2.8 (b)). Cache KV perlu menyimpan kunci dan nilai untuk semua $[1] \tilde{y}$ kepala ini, yaitu, $\{(K_{\tilde{y}}, V_{\tilde{y}}, q_{\tilde{y}}), \dots, (K_{\tilde{y}}, V_{\tilde{y}}, q_{\tilde{y}})\}$.

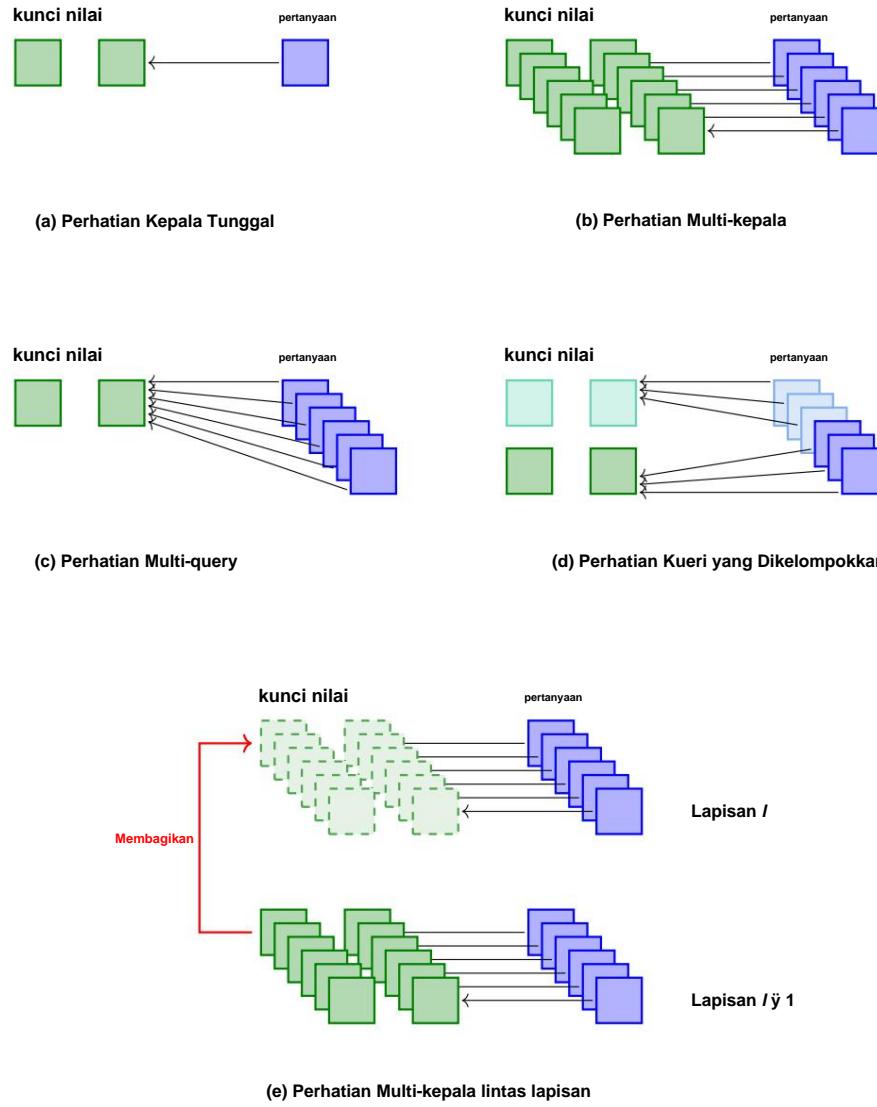
Salah satu penyempurnaan pada model perhatian multi-head, yang disebut perhatian multi-query (MQA), adalah dengan berbagi kunci dan nilai di seluruh head, sekaligus memungkinkan kueri menjadi unik untuk setiap head [Shazeer, 2019]. Dalam MQA, ada satu set kunci dan nilai ($K_{\tilde{y}}, V_{\tilde{y}}$). Selain itu, ada kueri q $\{q_{\tilde{y}}, \dots, q_{\tilde{y}}\}$, masing-masing sesuai dengan kepala yang berbeda. Untuk setiap kepala, kita memiliki

$$\text{kepalaj} = \text{Attqkv}(q, K_{\tilde{y}}, V_{\tilde{y}}) \quad (2.72)$$

Gambar 2.8 (c) mengilustrasikan model ini. Dengan berbagi kunci dan nilai, ukuran cache KV akan

2.3 Pemodelan Urutan Panjang

81



Gambar 2.8: Ilustrasi perhatian QKV berdasarkan mekanisme multi-head dan berbagi yang berbeda. (a) = single-head perhatian, dan (menjadi) = perhatian dengan banyak kepala.

jadilah $O(L \cdot dh \cdot m)$.

Perhatian kueri yang dikelompokkan (GQA) merupakan perluasan alami dari perhatian multi-kepala dan MQA [Ainslie et al., 2023]. Dalam GQA, kepala dibagi menjadi beberapa kelompok, yang masing-masing sesuai dengan tujuan bersama. kumpulan kunci dan nilai. Oleh karena itu kita memiliki ng kumpulan kunci dan nilai $\{(K^{[1]}_{ysaya, ysaya}, V^{[1]}_{ysaya, ysaya}), \dots, (K^{[id]}_{ysaya, ysaya}, V^{[id]}_{ysaya, ysaya})\}$. Lihat Gambar 2.8 (d) untuk ilustrasi. Misalkan $g(j)$ adalah id grup untuk kepala j -th. Model GQA dapat diungkapkan sebagai

$$\text{kepalaj} = \text{Attqkv}(q, \underbrace{\dots, \underbrace{[g(j)]}_{ysaya, ysaya}, \underbrace{[g(j)]}_{ysaya, ysaya}}_{ysaya, ysaya}) \quad (2.73)$$

Ukuran cache KV dari GQA adalah $O(L \cdot ng \cdot dh \cdot m)$. Salah satu manfaat GQA adalah kita dapat melakukan trade-off antara efisiensi komputasi dan ekspresivitas model dengan menyesuaikan ng . Ketika $ng = j$ model menjadi model perhatian multi-kepala standar. Sebaliknya, ketika $ng = 1$, itu menjadi

model GQA.

Pembagian juga dapat dilakukan lintas lapisan. Metode seperti itu termasuk dalam keluarga metode bobot bersama dan metode aktivasi bersama, yang telah banyak digunakan dalam Transformers [Dehghani et al., 2018; Lan et al., 2020]. Misalkan, seseorang dapat berbagi aktivasi KV atau bobot perhatian lintas lapisan untuk mengurangi jejak komputasi dan memori [Xiao et al., 2019; Brandon et al., 2024].

Gambar 2.8 (e) menunjukkan ilustrasi metode ini, di mana kueri dalam satu lapisan langsung mengakses cache KV pada lapisan di bawahnya.

2.3.5 Ekstrapolasi dan Interpolasi Posisi

Karena lapisan Transformer tidak peka terhadap urutan masukan, kita memerlukan beberapa cara untuk mengodekan informasi posisional dalam token masukan. Untuk melakukannya, biasanya ditambahkan penempatan posisional ke penempatan token, lalu memasukkan penempatan gabungan ini ke dalam tumpukan lapisan Transformer sebagai masukan. Dalam kasus ini, penyematan pada posisi i dapat dinyatakan sebagai

$$e_i = x_i + PE(i) \quad (2.74)$$

di mana $x_i \in \mathbb{R}^D$ menunjukkan penyematan token, dan $PE(i) \in \mathbb{R}^D$ menunjukkan penempatan posisi. Secara umum, penyematan token x_i adalah vektor yang tidak bergantung pada posisi, sehingga penyematan posisi $PE(i)$ digunakan untuk mengodekan konteks posisi. Pendekatan yang mudah adalah dengan memperlakukan $PE(i)$ sebagai variabel yang dapat dipelajari dan melatihnya bersama parameter model lainnya. Dengan cara ini, kita dapat mempelajari representasi unik untuk setiap posisi, dan dengan demikian membedakan token yang muncul pada posisi yang berbeda dari suatu urutan.

Representasi posisi menggunakan vektor yang dipelajari dapat berfungsi dengan baik dalam tugas-tugas yang urutannya pada waktu pelatihan dan pengujian memiliki panjang yang sama. Namun, dalam praktiknya, kami sering kali memberlakukan pembatasan panjang pada urutan selama pelatihan untuk mencegah biaya komputasi yang berlebihan, tetapi ingin menerapkan model yang dilatih ke urutan yang jauh lebih panjang selama inferensi. Dalam kasus ini, penggunaan penempatan posisi yang dipelajari memiliki kekurangan yang jelas, karena tidak ada penempatan yang dilatih untuk posisi yang tidak diamati dalam fase pelatihan.

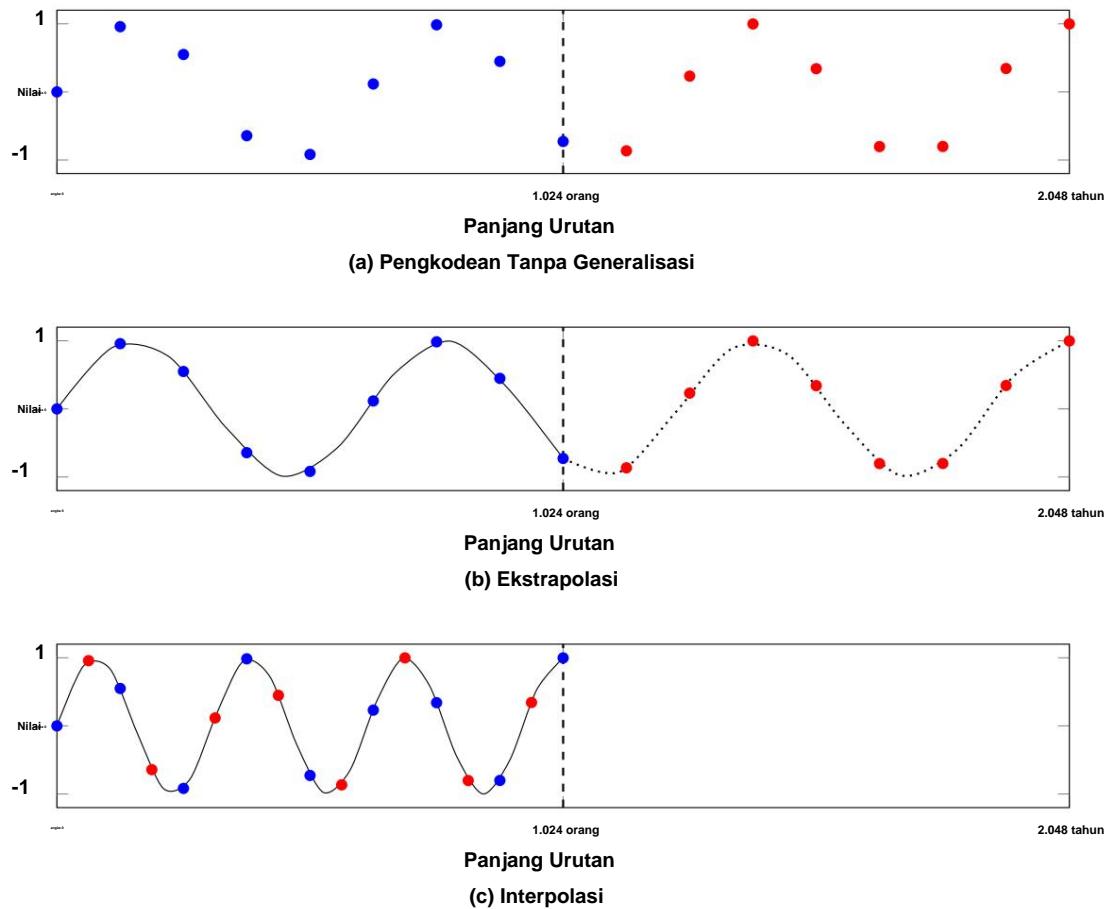
Pendekatan alternatif untuk memodelkan informasi posisi adalah mengembangkan penempatan posisi yang dapat digeneralisasi: setelah dilatih, model penempatan dapat digunakan untuk menangani sekuens yang lebih panjang. Misalkan kita melatih model penempatan posisi pada sekuens dengan panjang maksimum m_l , dan kita ingin menerapkan model yang dilatih ke sekuens dengan panjang m ($m >> m_l$). Jika model penempatan dibatasi dalam rentang posisi yang dapat kita amati dari data pelatihan, maka model ini akan gagal menangani data baru di luar rentang tersebut. Lihat Gambar 2.9 (a) untuk ilustrasi di mana model penempatan yang dipelajari tidak dapat memodelkan titik data di luar domain pelatihan jika tidak memiliki kemampuan untuk mengekstrapolasi.

Ada beberapa pendekatan untuk menggeneralisasi model penempatan posisi. Pendekatan-pendekatan tersebut dapat dikelompokkan menjadi dua kelas.

- **Ekstrapolasi.** Model yang dipelajari pada titik data yang diamati (yaitu, posisi) dapat langsung digunakan untuk menetapkan nilai yang berarti pada titik data di luar rentang asli. Misalkan, anggaplah kita memiliki serangkaian angka 1, 2, ..., 10, dan kita ingin memahami arti dari angka baru, 15. Mengetahui bahwa angka-angka ini adalah angka asli yang digunakan untuk pengurutan, kita dapat dengan mudah menyimpulkan bahwa 15 adalah angka yang mengikuti 10, meskipun 15 belum

2.3 Pemodelan Urutan Panjang

83



Gbr. 2.9: Ilustrasi metode penempatan posisi yang berbeda untuk berbagai posisi. Titik biru menunjukkan posisi yang telah diamati selama pelatihan, dan titik merah mewakili posisi yang baru diamati pada saat pengujian waktu. Pada sub-gambar (a), model pengkodean hanya mengingat titik-titik yang terlihat selama pelatihan, dan tidak dapat digeneralisasi. sub-gambar (b) dan (c), model dapat digeneralisasi melalui ekstrapolasi dan interpolasi.

telah diamati sebelumnya. Gambar 2.9 (b) menunjukkan contoh pendekatan ini, dimana suatu fungsi dipelajari untuk menyesuaikan titik data dalam rentang tertentu dan kemudian diterapkan untuk memperkirakan nilai titik data di luar rentang tersebut.

- **Interpolasi.** Pendekatan ini memetakan rentang titik data yang lebih besar ke dalam rentang observasi asli.

Misalnya, anggaplah kita memiliki model yang dirancang untuk angka dalam rentang $[1, 10]$. Ketika diberikan rentang baru $[1, 20]$, kita dapat memperkecilnya dengan membagi setiap angka dengan 2, sehingga semua angka masuk ke dalam $[1, 10]$. Skala ini memungkinkan kita untuk menggunakan model dilatih pada rentang $[1, 10]$ untuk menggambarkan titik data dalam rentang yang diperluas $[1, 20]$. Lihat Gambar 2.9 (c) untuk ilustrasi pendekatan ini.

Faktanya, penempatan posisi dalam banyak sistem telah mencapai beberapa tingkat generalisasi. Misalnya, pengkodean sinusoidal, metode penyematan posisi yang paling umum, menggunakan sinus dan fungsi kosinus yang secara alami dapat diperluas ke urutan dengan panjang apa pun. Meskipun pendekatan ini mungkin tampak langsung dan sederhana, namun tidak bekerja dengan baik ketika kita memperluas urutan secara signifikan untuk diproses. Pada subbagian ini, kita akan membahas beberapa metode alternatif berdasarkan ekstrapolasi atau interpolasi.

2.3.5.1 Perhatian dengan Bias yang Dapat Dipelajari

Satu masalah dengan Persamaan (2.74) adalah bahwa model penyisipan memperlakukan setiap token secara independen dan karenanya mengabaikan jarak antara token yang berbeda. Perbaikan umum untuk model ini, yang disebut penyisipan posisi relatif, adalah dengan mempertimbangkan hubungan berpasangan antara token [Shaw et al., 2018]. Ide umum di balik ini adalah untuk memperoleh offset antara pasangan posisi apa pun dan memasukkannya ke dalam model self-attention. Salah satu bentuk self-attention paling sederhana dengan penyisipan posisi relatif diberikan oleh

$$\text{Att}qkv(q_i, K_j, V_j) = \sum_{j=0}^d \hat{y}(i, j)v_j \quad (2.75)$$

$$\hat{y}(i, j) = \text{Softmax}\left(\frac{\mathbf{q}_i \mathbf{k}_j^T + \text{PE}(i, j)}{d}\right) + \text{Masker}(i, j) \quad (2.76)$$

Satu-satunya perbedaan antara model ini dan model self-attention asli adalah bahwa istilah bias $\text{PE}(i, j)$ ditambahkan ke produk kunci kueri dalam model baru ini. Secara intuitif, $\text{PE}(i, j)$ dapat diinterpretasikan sebagai penalti jarak untuk pasangan posisi i dan j . Saat i menjauh dari j , nilai $\text{PE}(i, j)$ menurun.

$\text{PE}(i, j)$ dapat didefinisikan dengan beberapa cara berbeda. Di sini, kami mempertimbangkan versi T5 dari penyematan posisi relatif, yang disebut bias T5 [Raffel et al., 2020]. Untuk setiap pasangan kueri q_i dan kunci ¹⁵ k_j , offset di antara keduanya didefinisikan

$$d(i, j) = i - j \quad (2.77)$$

Desain sederhana untuk bias $\text{PE}(i, j)$ adalah dengan menggunakan variabel yang dapat dipelajari yang sama untuk semua pasangan kunci kueri dengan offset yang sama, yaitu, $\text{PE}(i, j) = u_{ij}$, di mana u_{ij} adalah variabel yang sesuai dengan offset $i - j$. Namun, hanya dengan menetapkan nilai unik untuk setiap offset akan membatasi model ini pada offset yang diamati. Ketika $i - j$ lebih besar dari offset maksimum yang dilatih, model tidak dapat digeneralisasi.

Bias T5 mengadopsi generalisasi dari model ini. Alih-alih menetapkan setiap offset kunci kueri dengan istilah bias yang unik, ia mengelompokkan offset perbedaan ke dalam "bucket", yang masing-masing sesuai dengan satu parameter yang dapat dipelajari. Lebih khusus lagi, istilah bias untuk bucket $nb + 1$ diberikan sebagai berikut.

Untuk bucket 0 hingga $\lfloor \frac{nb+1}{2} \rfloor$, setiap bucket sesuai dengan satu offset, yaitu, bucket 0 untuk offset 0, bucket 1 untuk offset 1, bucket 2 untuk offset 2, dan seterusnya.

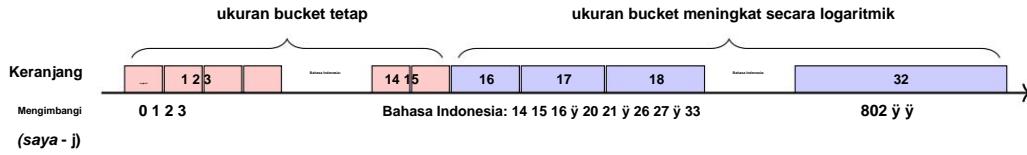
Kita nyatakan ini sebagai $b(i - j) = i - j$.

- Untuk bucket $\lfloor \frac{nb+1}{2} \rfloor$ hingga nb , ukuran setiap bucket meningkat secara logaritmik. Misalnya, nomor bucket untuk offset $i - j$ yang diberikan dapat didefinisikan sebagai $\frac{\log(i - j)}{\log(nb+1)}$

$$b(i - j) = \frac{1 + \frac{\log(i - j)}{\log(nb+1)}}{2} \quad (2.78)$$

di mana parameter $distmax$ biasanya diatur ke angka yang relatif besar untuk menunjukkan

¹⁵Untuk pemodelan bahasa, sebuah kueri hanya diperbolehkan memperhatikan konteks kirinya, sehingga kita memiliki $i > j$. Dalam kasus perhatian-diri yang lebih umum, di mana sebuah token dapat memperhatikan semua token dalam urutan tersebut, kita mungkin memiliki offset negatif ketika $i < j$.



Gbr. 2.10: Ilustrasi pendistribusian offset kunci kueri ke dalam bucket dalam model T5 ($nb = 32$ dan $distmax = 1024$). Kotak mewakili ember. Pada paruh pertama ember, kami menggunakan ukuran ember yang tetap. Pada paruh kedua ember, kami meningkatkan ukuran bucket secara logaritmik. Bucket terakhir berisi semua offset kunci kueri yang tidak tercakup oleh ember sebelumnya.

offset maksimum yang mungkin kita temui.

- Bila $i \leq j > distmax$, kita tempatkan $i \leq j$ di bucket terakhir. Dengan kata lain, bucket nb berisi semua offset yang tidak ditetapkan ke bucket sebelumnya.

Bersama-sama, hal ini dapat dinyatakan sebagai fungsi

$$b(i \leq j) = \begin{cases} \frac{saya - j}{2^{\lceil \log_2(\text{catatan}, \frac{tidak\ ada+1}{2}) \rceil}} + \frac{\log(i \leq j) \log(\frac{tidak\ ada+1}{2})}{\log(distmax) \log(\frac{tidak\ ada+1}{2})} & 0 \leq i \leq j < 2 \\ \frac{tidak\ ada+1}{2} & \text{tidak ada+1} \end{cases} \quad (2.79)$$

Gambar 2.10 menunjukkan ilustrasi bucket ini. Kita melihat bahwa di paruh pertama bucket, setiap bucket dikaitkan hanya dengan satu nilai $i \leq j$, sedangkan pada bagian kedua, ukuran bucket meningkat seiring pertumbuhan $i \leq j$. Bucket terakhir dirancang untuk menangani urutan panjang yang sembarang panjangnya.

Semua $PE(i, j)$ dalam bucket memiliki suku bias yang sama $ub(i \leq j)$. Substitusikan $PE(i, j) = ub(i \leq j)$ ke persamaan (2.76) bobot perhatian untuk qi dan kj menjadi 16

$$j \leq(i, j) = \text{Softmax}\left(\frac{\text{cepat}^T + ub(i \leq j)}{d} + \text{Masker}(i, j)\right) \quad (2.81)$$

Parameter $\{u_0, \dots, unb\}$ dipelajari sebagai parameter umum selama pelatihan. Perlu ditekankan bahwa model ini dapat digeneralisasi ke sekvens yang panjang. Hal ini karena $PE(i, j)$ s dengan offset kunci kueri yang serupa berbagi parameter yang sama, dan strategi berbagi ini sangat penting untuk mencapai generalisasi yang baik, mengingat offset kunci kueri yang besar jarang terjadi dalam pelatihan. Dalam praktiknya, kita sering menetapkan nb ke angka sedang, sehingga dapat membantu mengendalikan overfitting model penanaman posisional.

¹⁶Perlu dicatat bahwa, dalam model T5 Raffel et al. [2020], operasi penskalaan ulang untuk produk kunci kueri dihilangkan. bobot perhatian $j(i, j)$ kemudian diberikan oleh

$$j(i, j) = \text{Softmax}(qik^T + ub(i \leq j) + \text{Masker}(i, j)) \quad (2.80)$$

2.3.5.2 Perhatian dengan Bias yang Tidak Dipelajari

Model penempatan posisi relatif didasarkan pada serangkaian bias yang dipelajari untuk produk kunci kueri dalam perhatian diri. Pendekatan alternatif adalah memberi bias ini nilai tetap melalui heuristik, daripada melatihnya pada kumpulan data tertentu. Salah satu manfaat dari pendekatan berbasis heuristik ini adalah pendekatan ini tidak bergantung pada proses pelatihan dan dengan demikian dapat langsung diterapkan ke urutan apa pun setelah bias ditetapkan.

Salah satu contoh pendekatan tersebut adalah pendekatan [Press et al. \[2022\]](#), yang disebut perhatian dengan bias linier atau disingkat ALiBi. Dalam pendekatan ALiBi, istilah bias didefinisikan sebagai offset kunci kueri berskala negatif

$$PE(i, j) = \hat{y} \cdot (i \hat{y} j) = \hat{y} \cdot (j \hat{y} i) \quad (2.82)$$

di mana \hat{y} adalah faktor skala. Dengan menambahkan istilah ini ke produk kunci kueri, kita memperoleh bentuk bobot perhatian baru

$$\hat{y}(i, j) = \text{Softmax}\left(\frac{qik^T + \hat{y} \cdot (j \hat{y} i)}{d} + \text{Masker}(i, j)\right) \hat{y} \quad (2.83)$$

Model ini dapat diartikan sebagai penambahan penalti tetap pada qik^T setiap kali j bergerak satu langkah menjauh dari i . Jadi kita tidak perlu mengadaptasinya ke berbagai panjang sekvens, dan dapat menggunakan untuk memodelkan sekvens yang panjangnya sembarang. Lihat Gambar 2.11 untuk perbandingan bias T5 dan bias ALiBi.

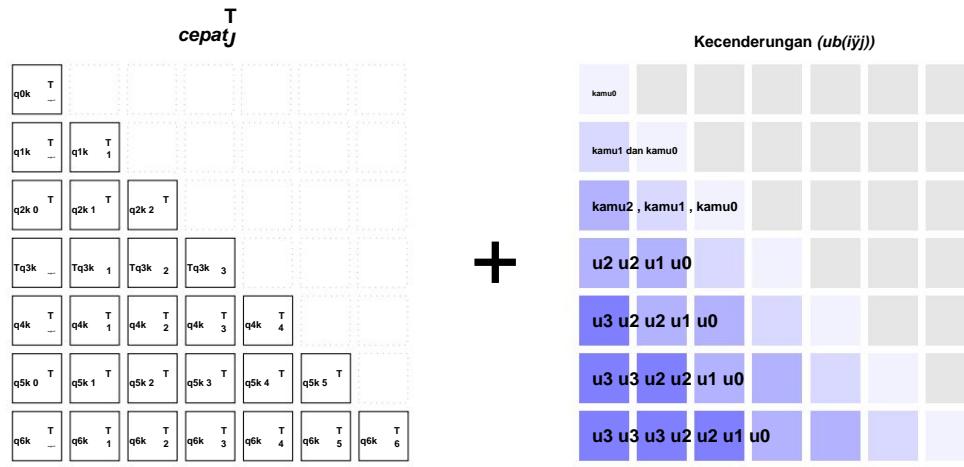
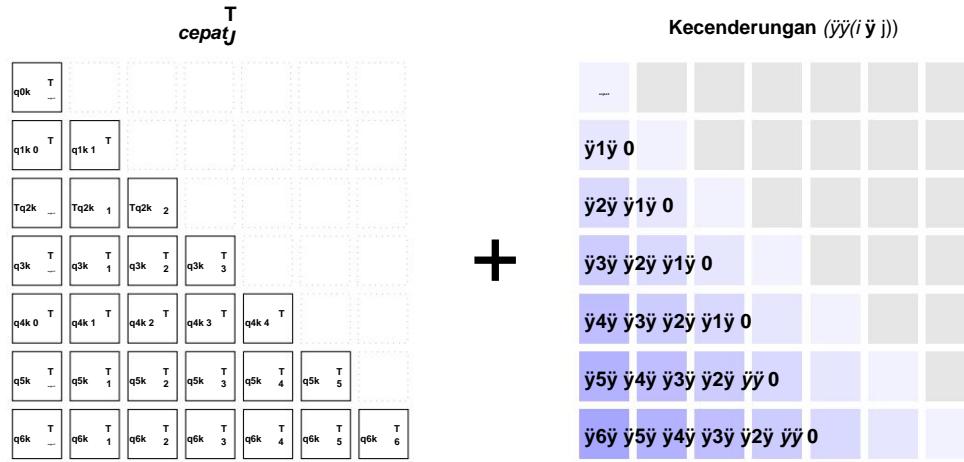
Secara umum, skalar \hat{y} harus disetel pada kumpulan data validasi. Namun, [Press et al. \[2022\]](#) untuk menemukan bahwa pengaturan \hat{y} ke nilai yang menurun secara geometris dengan faktor perhatian multi-kepala bekerja dengan baik pada berbagai tugas. Secara khusus, untuk sub-lapisan self-attention yang melibatkan n_{head} , skalar untuk head ke- k diberikan oleh

$$\hat{y}_k = \frac{1}{2^{\frac{s}{n_{\text{head}}}}} \quad (2.84)$$

Pendekatan ALiBi menyediakan bentuk sederhana dari penempatan posisi relatif. Ada metode serupa lainnya untuk merancang bias kunci kueri menggunakan offset $i \hat{y} j$. Tabel 2.4 menunjukkan perbandingan bias tersebut. Sebagai tambahan, perlu dicatat bahwa bentuk sisi kanan Persamaan (2.82) sangat mirip dengan fitur panjang yang digunakan dalam sistem berbasis fitur konvensional. Misalnya, dalam sistem penerjemahan mesin statistik, fitur tersebut banyak digunakan untuk memodelkan masalah penataan ulang kata, yang menghasilkan model yang dapat digeneralisasi dengan baik di berbagai tugas penerjemahan [\[Koehn, 2010\]](#).

2.3.5.3 Penanaman Posisi Putar

Seperti halnya penyematan sinusoidal, penyematan posisi putar didasarkan pada nilai-nilai yang dikodekan secara kaku untuk semua dimensi penyematan [\[Su et al., 2024\]](#). Ingat kembali bahwa dalam model penyematan sinusoidal, posisi direpresentasikan sebagai kombinasi fungsi sinus dan kosinus dengan frekuensi yang berbeda. Penanaman ini kemudian ditambahkan ke penanaman token untuk membentuk masukan ke Transformer

(a) Bias T5 ($nb = 3$ dan $distmax = 5$)

(b) Bias ALiBi

Gambar 2.11: Produk kunci kueri dengan bias (atas = bias T5 dan bawah = bias ALiBi). Skala warna bias berkisar dari biru muda yang menunjukkan nilai absolut kecil hingga biru tua yang menunjukkan nilai absolut besar.

tumpukan lapisan. Penanaman posisi putar memodelkan konteks posisi sebagai rotasi ke penanaman token dalam ruang kompleks. Hal ini mengarah ke model yang dinyatakan dalam bentuk penanaman perkalian

$$\text{nilai } ei = xiR(i) \quad (2.85)$$

di mana $R(i) \in \mathbb{R}^{D \times D}$ adalah matriks rotasi yang mewakili rotasi yang dilakukan pada token penanaman $xi \in \mathbb{R}^D$.

Untuk menyederhanakannya, pertama-tama kita akan mempertimbangkan penyemat dengan hanya dua dimensi dan kembali ke pembahasan tentang formulasi yang lebih umum nanti. Misalkan kita memiliki penyemat token 2 dimensi $x = x_1 x_2$. Kita dapat merepresentasikannya sebagai vektor dalam bidang, yang berawal dari titik asal $(0, 0)$ dan berakhir di (x_1, x_2) . Rotasi berlawanan arah jarum jam dari vektor ini mengacu pada

Entri	Bias Kunci Kueri ($\text{PE}(i, j)$)
T5 [Raffel dkk., 2020] $ub(i\bar{y}j)$	
ALiBi [Press et al., 2022] $\bar{y} \cdot (i \bar{y} j) \bar{y}^1 (i \bar{y} j) \bar{y}^2$	
Kerple [Chi et al., 2022] \bar{y}^1	(daya)
	$\log(1 + \bar{y}^2(i \bar{y} j))$ (logaritmik) $\cos(i \bar{y} j) / 100002k/d^- f(\bar{y}(i$
Roti lapis [Chi dkk., 2023]	d^- $2 \sum_{k=1}^{m-1} \bar{y}^k / \bar{y}(\max(mlen, i))$
KEBAKARAN [Li et al., 2024]	

Tabel 2.4: Bias kunci kueri sebagai penempatan posisi relatif. \bar{y} , \bar{y}^1 , \bar{y}^2 , d^- , dan mlen adalah hiperparameter. Dalam model T5, $b(i \bar{y} j)$ menunjukkan bucket yang ditetapkan ke $i \bar{y} j$. Dalam model FIRE, $\bar{y}(\cdot)$ adalah fungsi yang meningkat secara monoton seperti $\bar{y}(x) = \log(cx + 1)$, dan $f(\cdot)$ adalah FFN.

menggerakkan vektor di sekitar titik asal sambil mempertahankan besarnya, seperti yang ditunjukkan pada Gambar 2.12 (a). Derajat rotasi biasanya ditentukan oleh sudut tertentu, dilambangkan dengan \bar{y} . Rotasi dapat dinyatakan secara matematis dalam bentuk

$$\begin{aligned}
 & \text{Rumus untuk persamaan } Ro(x, \bar{y}) = xR\bar{y} \\
 & = x_1 x_2 \quad \frac{\cos \bar{y} \cos \bar{y}}{\sin \bar{y} \sin \bar{y}} \\
 & = \cos \bar{y} \cdot x_1 \bar{y} \sin \bar{y} \cdot x_2 \sin \bar{y} \cdot x_1 + \cos \bar{y} \cdot x_2 \quad (2.86)
 \end{aligned}$$

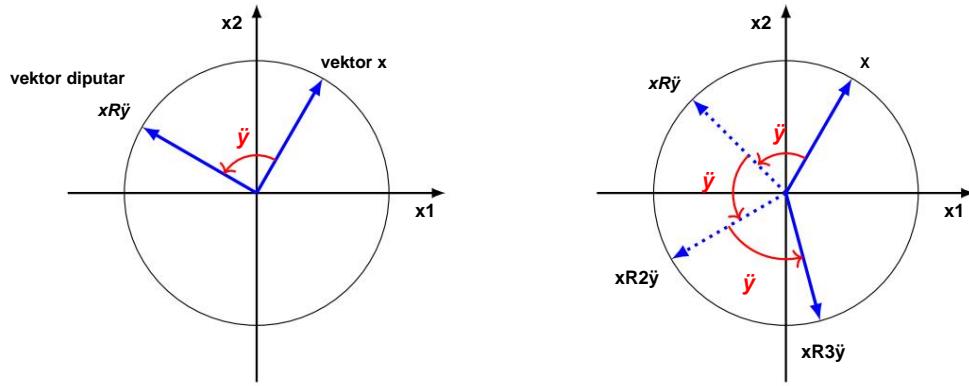
dimana $R\bar{y} = \begin{pmatrix} \cos \bar{y} & -\sin \bar{y} \\ \sin \bar{y} & \cos \bar{y} \end{pmatrix}$ adalah matriks rotasi. Jika dua atau lebih rotasi dilakukan pada vektor yang sama, kita dapat memutar vektor lebih jauh. Hal ini mengikuti fakta bahwa komposisi rotasi berturut-turut itu sendiri adalah rotasi. Secara lebih formal, memutar vektor dengan sudut \bar{y} sebanyak t kali dapat dinyatakan sebagai

Rumus untuk persamaan $Ro(x, t\bar{y})$ adalah :

$$= biaya\bar{y} \cdot x_1 \bar{y} \sin t\bar{y} \cdot x_2 \sin t\bar{y} \cdot x_1 + biaya\bar{y} \cdot x_2 \quad (2.87)$$

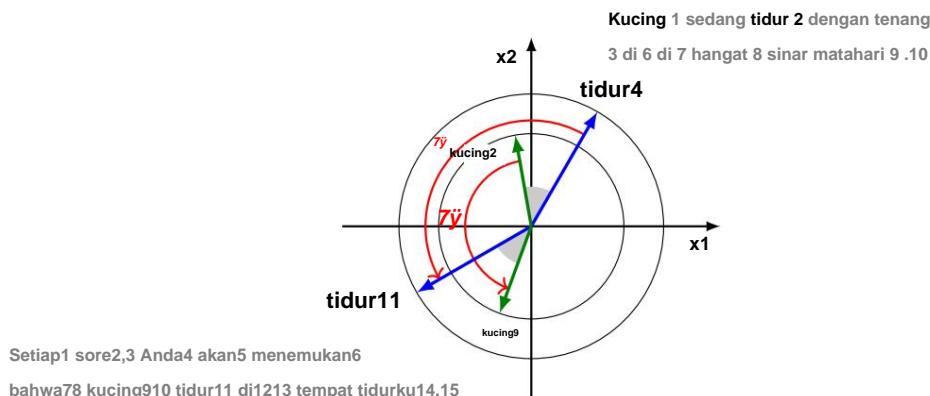
Jika kita mengartikan t sebagai posisi token yang direpresentasikan oleh x dalam suatu urutan, maka kita akan menemukan bahwa persamaan di atas mendefinisikan model penempatan posisional yang sederhana. Seperti yang ditunjukkan pada Gambar 2.12 (b), kita mulai menggerakkan token dari posisi 0. Setiap kali kita bergerak satu langkah ke depan, vektor diputar oleh sudut \bar{y} . Setelah tiba di posisi t , representasi token dengan konteks posisional diberikan oleh $Ro(x, t\bar{y})$. Karena rotasi tidak mengubah besarnya penempatan, "makna" asli token dipertahankan. Informasi posisional disuntikkan ke dalam penempatan, saat diputar.

Cara populer untuk memahami rotasi vektor adalah dengan mendefinisikannya dalam ruang kompleks. Mudah untuk mengubah setiap vektor $x = x_1 x_2$ dalam ruang Euclidean 2D R^2 untuk menemukan bilangan kompleks $x = x_1 + ix_2$ dalam ruang kompleks C melalui pemetaan linier bijektif. Kemudian, rotasi x sudut $t\bar{y}$ sesuai dengan perkalian dengan $e^{it\bar{y}}$. Diketahui bahwa $e^{it\bar{y}}$ dengan $= \cos \bar{y} + i \sin \bar{y}$, rotasi



(a) Rotasi Satu Langkah

(b) Rotasi Multi-langkah



(c) Sudut antara penempatan dua token pada posisi yang berbeda

Gbr. 2.12: Ilustrasi rotasi vektor pada bidang. Subgambar (a) dan (b) menunjukkan rotasi vektor dalam satu langkah dan beberapa langkah berturut-turut. Subgambar (c) menunjukkan penempatan token cat dan sleeping dalam dua kalimat yang berbeda. Kami menunjukkan kalimat-kalimat ini dengan subskrip yang dibubuhkan pada setiap token untuk menunjukkan posisinya. Jika kami merepresentasikan token sebagai vektor, kami dapat menambahkan informasi posisi dengan memutar vektor ini. Rotasi ini mempertahankan "jarak" antara vektor. Misalnya, mengingat jarak antara cat dan sleeping sama dalam kedua kalimat, sudut antara penempatannya juga tetap sama selama rotasi.

operasi dapat diungkapkan kembali dalam bentuk

$$\begin{aligned}
 & xRtÿ \bar{y} x \\
 & = (x1 + ix2)(biaya\bar{y} + i \sin tÿ) = biaya\bar{y} \\
 & \quad \cdot x1 \bar{y} \sin tÿ \cdot x2 + i(\sin tÿ \cdot x1 + biaya\bar{y} \cdot x2)
 \end{aligned} \tag{2.88}$$

Di sini kita menunjukkan representasi token $x \bar{y} e^{itÿ}$ oleh $C(x, tÿ)$. Produk dalam dari representasi tokens dari token pada posisi t dan s dapat ditulis sebagai

$$\text{Persamaan } (x, tÿ) \text{ dan } (y, sÿ) = (x \bar{y} \bar{y}) e^{i(tÿ-sÿ)} \tag{2.89}$$

di mana $\bar{y} \bar{y}$ adalah konjugat kompleks dari y . Seperti yang dapat dilihat, hasil dari produk dalam ini melibatkan suku $y tÿ s$, sehingga dapat memodelkan pergeseran antara dua token.

Sekarang kita kembali ke representasi dalam ruang Euclidean 2D. Perkalian titik dari $\text{Ro}(x, t\ddot{y})$ dan $\text{Ro}(y, s\ddot{y})$ dapat ditulis sebagai fungsi dari $(t\ddot{y} \ s\ddot{y})$

$$\begin{aligned} \text{Rumus } \text{Ro}(x, t\ddot{y})[\text{Ro}(y, s\ddot{y})]^T &= x \text{R}t\ddot{y}[y \text{R}s\ddot{y}]^T \\ &= x \text{R}(t\ddot{y}s\ddot{y})^T \end{aligned} \quad (2.90)$$

Mengingat hasil ini, jika kita menganggap $\text{Ro}(x, t\ddot{y})$ dan $\text{Ro}(y, s\ddot{y})$ sebagai kunci dan kuncinya, maka operasi perhatian-diri secara implisit akan melibatkan pemodelan konteks posisi relatif.

Penanaman posisi putar ini dapat diperluas ke penanaman multidimensi. Untuk d -dimensi 2 penanaman token d -dimensi $x = x_1 \ x_2 \dots x_d$, kita dapat memperlakukannya sebagai vektor kompleks $x = x_1 + i x_2 + \dots + i x_d$, mana $= x_1 + i x_2 x_3 + i x_4 \dots x_d \ddot{y}_1 + i x_d$, setiap pasangan item yang berurutan membentuk bilangan kompleks. Kemudian, penempatan posisi putar dalam ruang kompleks diberikan oleh

$$\begin{gathered} \text{Tentukan nilai } x \text{ dan } t\ddot{y} \quad \sum_{k=1}^{d/2} x_k e^{i t\ddot{y}_k} \\ \text{hari ke } 2 \end{gathered} \quad (2.91)$$

di mana e_k adalah vektor basis standar dengan satu nilai bukan nol di koordinat ke- k dan 0 di tempat lain [Biderman et al., 2021].

Meskipun rumus ini melibatkan ekspresi yang rumit, bentuk ekuivalennya dalam d -dimensi Ruang Euclidean relatif mudah dipahami. Kita dapat menuliskannya sebagai

$$\begin{gathered} \text{Rumus untuk persamaan } \text{Ro}(x, t\ddot{y}) \text{ adalah } x_1, x_2, \dots, x_d. \\ \begin{array}{c} \text{R}t\ddot{y}_1 \\ \vdots \\ \text{R}t\ddot{y}_2 \\ \vdots \\ \text{R}t\ddot{y}_{d/2} \end{array} \end{gathered} \quad (2.92)$$

dimana $\text{R}t\ddot{y}_k = \frac{\cos t\ddot{y}_k \sin t\ddot{y}_k \cdot \ddot{y}}{\cos t\ddot{y}_k} = \ddot{y}_1, \dots, \ddot{y}_{d/2}$ adalah parameter untuk mengendalikan arah sin $t\ddot{y}_k$ sudut rotasi dalam dimensi yang berbeda. Biasanya, \ddot{y}_k ditetapkan ke $10000 \frac{\text{hari}}{2(k-1)}$ yang analog ke pengaturan dalam penyematan sinusoidal.

Dalam implementasi praktis, Persamaan (2.92) dapat ditulis ulang ke dalam bentuk yang hanya bergantung pada produk unsur-unsur dan penambahan vektor.

$$\begin{gathered} \text{Rumus } \text{Ro}(x, t\ddot{y}) = \begin{bmatrix} x_1 & \text{T} & \text{biaya} \ddot{y}_1 & \text{T} & -x_2 & \text{T} & \text{dosa} \ ddot{y}_1 & \text{T} \\ x_2 & \vdots & \text{biaya} \ddot{y}_1 & \vdots & x_1 & \vdots & \text{dosa} \ ddot{y}_1 & \vdots \\ \vdots & \ddot{y} & \vdots & + & \vdots & \ddot{y} & \vdots & \vdots \\ x_d \ddot{y}_1 & \text{biaya} \ddot{y}_d / & -x_d & \text{dosa} \ ddot{y}_d / \\ x_D & 2 \ddot{y} \text{ biaya} \ddot{y}_d / 2 & x_d \ddot{y}_1 & 2 \ddot{y} \text{ dosa} \ ddot{y}_d / 2 \end{bmatrix} \end{gathered} \quad (2.93)$$

Akhirnya, kita tulis ulang Persamaan (2.85) untuk memperoleh bentuk penanaman pada posisi i

$$ei = Ro(xi, ij) \quad (2.94)$$

2.3.5.4 Interpolasi Posisi

Dalam interpolasi posisi, tujuan kami adalah memetakan posisi dalam urutan baru agar sesuai dengan rentang yang diamati dalam pelatihan. Misalkan panjang urutan untuk pelatihan berkisar dari 0 hingga mI .

Ketika $m > mI$ pada waktu pengujian, kami merepresentasikan posisi dalam $[0, m]$ sehingga representasi kami sesuai dengan $[0, mI]$.

Sebagai ilustrasi, perhatikan model penempatan posisi putar yang dijelaskan di atas. Penempatan setiap token dijelaskan oleh model $Ro(xi, ij)$ di mana $\bar{y} = \bar{y}_1, \dots, \bar{y}_d/2$ adalah parameternya. $Ro(xi, ij)$ dapat dinyatakan dalam bentuk kombinasi linier dari dua fungsi periodik (lihat Persamaan (2.93)).

$$kosi\bar{y} = kosi\bar{y}_1 \dots kosi\bar{y}_d/2 \quad (2.95)$$

$$\text{dosa } ij = \text{dosa } ij_1 \dots \text{dosa } ij_d/2 \quad (2.96)$$

\bar{y}_k adalah fungsi eksponensial dari k dan berbentuk

$$\bar{y}_k = b \bar{y}^{\frac{2(k-1)}{2(k-1) \text{ hari}}} \quad (2.97)$$

dimana b adalah basis. Periode $\cos i\bar{y}_k$ dan $\sin i\bar{y}_k$ adalah

$$T_k = 2\bar{y} \cdot b^{\frac{2(k-1)}{2(k-1)}} \quad (2.98)$$

Ide utama di balik interpolasi posisi adalah untuk menyesuaikan periode ini sehingga posisi baru dapat dikodekan dalam kisaran $[0, mI]$. Salah satu cara untuk mencapai hal ini adalah dengan meningkatkan T_k dengan $\frac{M}{mI}$ diberikan oleh

$$T'_k = \frac{M}{mI} \cdot b^{\frac{2(k-1) \cdot 2\bar{y}}{2(k-1) \cdot 2\bar{y}}} \quad (2.99)$$

Oleh karena itu semua titik di $[0, m]$ dikompresi menjadi $[0, mI]$. Penskalaan linier ini dapat dengan mudah diwujudkan dengan memodifikasi input ke model penyematan [Chen et al., 2023c]. Model baru dengan interpolasi posisi linier diberikan oleh

$$\text{Rumus } Ro\bar{y}(xi, ij) = Ro(xi, \frac{mI}{M} \text{ (saya)}) \quad (2.100)$$

Metode interpolasi posisional lainnya adalah dengan menskalakan basis¹⁷. Misalkan basis b adalah diskalakan oleh \bar{y} . Kami berharap periode model baru ini dalam dimensi terakhir \bar{y} (yaitu, dimensi) sama $\frac{D}{2}$ dengan periode model interpolasi posisi linier. Hal ini dapat dinyatakan sebagai

$$2\bar{y} \cdot (\bar{y}b)^{\frac{d-1}{(2d-1)}} = \frac{M}{2\bar{y} \cdot b^{\frac{d}{2}} mI} \cdot \frac{\frac{d}{2}-1}{2(\frac{d}{2}-1)} \quad (2.101)$$

¹⁷Metode ini pertama kali diusulkan di https://www.reddit.com/r/LocalLLaMA/comments/14lz7j5/ntkaware_scaled_rope_allows_llama_models_to_have/

Dengan memecahkan persamaan ini, kita memperoleh

$$\begin{aligned}\ddot{y} &= \frac{M}{\frac{m}{\frac{m}{m}} \frac{\text{hari}}{\text{dij2}}} \\ &= \frac{M}{\frac{m}{m}} \frac{\text{hari}}{\text{dij2}}\end{aligned}\quad (2.102)$$

Ini memberikan model penyematan

$$\text{Rumus untuk persamaan Ro}(\mathbf{x}_i, \mathbf{i}\ddot{y}) = \text{Ro}(\mathbf{x}_i, i\ddot{y}) \quad (2.103)$$

Di mana

$$\ddot{y}^d = (\ddot{y}b) \ddot{y} - \frac{d}{\text{hari}} . (\ddot{y}b) \ddot{y} - \frac{2}{\text{hari}} \dots , (\ddot{y}b) \ddot{y} - \frac{\text{hari}\ddot{y}^2}{\text{hari}} \quad (2.104)$$

Perhatikan bahwa penskalaan basis menyediakan metode yang tidak seragam untuk penskalaan periode di berbagai dimensi \ddot{y} . Metode ini terbukti bermanfaat untuk memperluas LLM ke urutan yang lebih panjang, dan beberapa perbaikan telah dikembangkan [Peng et al., 2024; Ding et al., 2024].

2.3.6 Keterangan

Pada bagian ini, kami telah menyajikan berbagai metode untuk pemodelan bahasa konteks panjang. Kami menutup bagian ini dengan membahas beberapa isu menarik yang terkait dengan metode ini.

2.3.6.1 Kebutuhan Konteks Panjang

Salah satu tujuan akhir dari LLM konteks panjang adalah bahwa model-model ini dapat secara tepat mengodekan konteks tak terbatas. Apa yang disebut konteks tak terbatas lebih mengacu pada fakta bahwa LLM dapat terus-menerus membaca kata-kata. Hal ini memotivasi LLM yang dapat menangani konteks atau aliran data yang sangat panjang. Seperti yang dibahas dalam Bagian 2.3.3, adalah umum untuk menggunakan model memori berukuran tetap untuk memproses konteks yang terus berkembang. Banyak sistem seperti itu didasarkan pada arsitektur berulang atau variannya, karena secara inheren cocok untuk memodelkan masalah deret waktu di mana efek dari masukan masa lalu berlanjut tanpa batas. Cara lain untuk mencapai memori tak terbatas adalah dengan mengembangkan alternatif untuk model perhatian diri, misalnya, seseorang dapat menggunakan model perhatian ruang-berkelanjutan untuk mengodekan konteks, yang menghilangkan ketergantungan pada panjang

Ketika mempelajari LLM konteks panjang, wajar untuk bertanya-tanya mekanisme apa yang dapat menjelaskan penggunaan konteks panjang dalam pemodelan bahasa. Bisakah kita memampatkan representasi konteks tak terbatas ke dalam model berukuran relatif kecil? Apakah semua token konteks berguna untuk memprediksi token berikutnya? Bagaimana LLM mempersiapkan prediksi token saat melihat konteksnya? Bisakah kita mengetahui terlebih dahulu informasi kontekstual mana yang penting untuk prediksi? Jawaban umum untuk semua pertanyaan ini tidak jelas, tetapi mereka menginspirasi penelitian lanjutan tentang model yang dapat dijelaskan, dan beberapa hasil menarik telah ditemukan. Misalnya, Deletang et al. [2024] melakukan eksperimen ekstensif untuk menunjukkan bahwa LLM adalah kompresor dalam konteks yang kuat. Meskipun melihat model prediktif sebagai model kompresi telah lama dipelajari dalam pembelajaran mesin, hal itu juga memberikan wawasan tentang pemahaman kita tentang hukum penskalaan LLM. Pal et al. [2023] dan Wu et al. [2024] menyelidiki apakah fitur yang dipelajari hingga langkah saa

untuk memprediksi token pada langkah-langkah berikut. Perhatikan bahwa kebutuhan akan konteks panjang dalam pemodelan bahasa sangat bergantung pada masalah yang kita tangani. Masalah terkait adalah di mana menerapkan LLM dan cara mengevaluasinya. Misalnya, dalam tugas-tugas ringkas, kita mungkin hanya perlu menyaring dan fokus pada beberapa aspek utama teks, sementara dalam tugas-tugas seperti pengambilan kembali, kita perlu "menghafal" seluruh konteks sehingga informasi yang relevan dapat diakses. Kita akan membahas evaluasi masalah nanti di subbagian ini.

2.3.6.2 Pra-pelatihan atau Adaptasi LLM?

Pelatihan LLM memerlukan biaya komputasi yang signifikan. Meskipun mudah untuk melatih LLM pada data sekuens panjang, pelatihan menjadi sulit secara komputasi untuk set data besar. Merupakan praktik umum untuk melatih LLM terlebih dahulu pada set data umum, lalu mengadaptasinya dengan upaya penyempurnaan yang sederhana. Misalnya, LLM dengan penempatan posisi relatif atau putar dapat dilatih langsung pada data skala besar dalam fase prapelatihan. Sementara model yang dihasilkan mungkin menunjukkan beberapa kemampuan untuk mengekstrapolasi panjang dalam fase inferensi, mungkin lebih efektif untuk menyempurnakannya pada sekuens yang lebih panjang.

Idealnya, kami ingin melakukan pra-pelatihan LLM dengan arsitektur Transformer standar dan mengadaptasinya ke tugas-tugas baru. Hal ini memungkinkan kami untuk menggunakan banyak LLM siap pakai dan secara efisien mengadaptasinya untuk menangani urutan yang panjang. Namun, ketika arsitektur baru diadopsi, tampaknya tak terelakkan bahwa kami perlu melatih model-model ini dari awal. Hal ini menimbulkan kesulitan praktis untuk mengembangkan LLM konteks panjang, karena kami tidak dapat memanfaatkan model-model yang telah dilatih sebelumnya dan dikembangkan dengan baik dan sebagai gantinya harus melatihnya sendiri. Di sisi lain, fine-tuning masih merupakan cara yang efektif untuk mengadaptasi LLM dengan arsitektur tertentu yang berbeda dari yang ada dalam pra-pelatihan. Contohnya adalah model yang ditambah dengan memori eksternal. Dalam model-model ini, LLM yang telah dilatih sebelumnya bersifat tetap, dan fokusnya adalah pada bagaimana membuat LLM ini berkolaborasi dengan model-model memori. Dalam RAG, misalnya, adalah umum untuk melakukan fine-tuning LLM untuk meningkatkan penggunaan input yang ditambah dengan pengambilan kembali. Contoh lain dari penyempurnaan LLM untuk pemodelan konteks panjang adalah kita melatih LLM dengan model perhatian penuh, lalu menggantinya dengan model perhatian jarang dalam fase penyempurnaan. LLM yang telah dilatih sebelumnya memberikan nilai awal parameter model yang digunakan dalam model lain, lalu mod-

2.3.6.3 Mengevaluasi LLM Konteks Panjang

Mengevaluasi LLM konteks panjang penting, tetapi ini merupakan isu baru dalam NLP. Ide umumnya adalah, jika kita memasukkan konteks panjang ke dalam LLM, maka kita dapat memeriksa dari keluaran LLM apakah ia memahami seluruh konteks dan memanfaatkannya dalam memprediksi token berikutnya. Dalam penelitian NLP konvensional, evaluasi semacam itu sering kali ditujukan untuk menguji kemampuan model NLP dalam menangani ketergantungan jangka panjang. Namun, ukuran konteks yang digunakan dalam LLM terkini jauh lebih besar daripada yang digunakan dalam sistem NLP beberapa tahun lalu. Hal ini memotivasi para peneliti untuk mengembangkan tolok ukur dan metrik evaluasi baru untuk LLM konteks panjang.

Salah satu pendekatan adalah dengan menggunakan metrik kebingungan. Akan tetapi, meskipun tampak sederhana, metode ini cenderung lebih mencerminkan kemampuan LLM untuk memanfaatkan konteks lokal daripada konteks global. Oleh karena itu, ada godaan untuk mengembangkan metode evaluasi yang khusus untuk LLM konteks panjang. Metode yang populer mencakup berbagai tugas sintetis yang dibuat atau dimodifikasi secara artifisial

Data digunakan untuk mengevaluasi kapabilitas spesifik LLM konteks panjang. Dalam **tugas mencari jarum dalam tumpukan jerami**¹⁸ dan kunci sandi [Mohtashami dan Jaggi, 2024; Chen et al., 2023c], misalnya, LLM diharuskan mengidentifikasi dan mengekstrak informasi kecil yang relevan dari sejumlah besar teks yang diberikan. Asumsinya di sini adalah bahwa LLM dengan memori yang cukup harus mengingat bagian teks sebelumnya saat memproses informasi baru. Dengan demikian, LLM ini dapat memilih detail yang relevan, yang mungkin jarang dan tersembunyi di antara banyak informasi yang tidak relevan, dari teks. Sebagai alternatif, dalam tugas memori salinan (atau tugas salin singkatnya), LLM digunakan untuk mengulang teks input atau segmen tertentu beberapa kali. Tugas-tugas ini awalnya diusulkan untuk menguji sejauh mana model berulang dapat mempertahankan dan mengingat token yang terlihat sebelumnya [Hochreiter dan Schmidhuber, 1997; Arjovsky et al., 2016], dan telah diadopsi dalam mengevaluasi LLM terbaru [Bulatov et al., 2022; Gu dan Dao, 2023].

Pendekatan lain untuk mengevaluasi LLM konteks panjang adalah mengujinya pada tugas NLP yang melibatkan urutan input yang sangat panjang. Contohnya termasuk ringkasan dokumen panjang atau multidokumen, menjawab pertanyaan dokumen panjang, pelengkapan kode, dan sebagainya. Manfaat dari pendekatan ini adalah dapat menyelaraskan evaluasi dengan harapan pengguna.

Meskipun banyak metode telah dikembangkan, masih belum ada cara umum untuk mengevaluasi LLM konteks panjang [Liu et al., 2024c]. Salah satu masalahnya adalah sebagian besar metode ini berfokus pada aspek-aspek tertentu dari LLM, daripada kemampuan mendasarnya untuk memodelkan konteks yang sangat panjang. Meskipun LLM dapat memilih bagian teks yang tepat dari masukan, kita tidak dapat mengatakan bahwa ia benar-benar memahami seluruh konteks. Sebaliknya, ia mungkin hanya mengingat beberapa bagian penting dari konteks, atau bahkan sekadar mengingat jawaban melalui model yang dipelajari dalam pra-pelatihan. Selain itu, data yang digunakan dalam banyak tugas berskala kecil dan relatif awal, yang menyebabkan perbedaan antara hasil evaluasi dan kinerja aplikasi aktual. Masalah yang lebih menarik adalah bahwa hasil LLM dipengaruhi oleh banyak faktor dan pengaturan eksperimen lainnya, misalnya, menggunakan perintah yang berbeda dapat menghasilkan hasil yang sangat berbeda. Hal ini membuat evaluasi menjadi lebih menantang karena perbaikan mungkin tidak hanya dihasilkan dari pemodelan konteks panjang yang lebih baik, dan ada risiko mengklaim hasil kami secara berlebihan. Meskipun demikian, masih banyak pertanyaan terbuka dalam pengembangan dan evaluasi LLM konteks panjang. Misalnya, model-model ini masih memiliki keterbatasan seperti panjang konteks yang terbatas dan latensi yang tinggi. Mempelajari masalah-masalah ini kemungkinan akan memberikan arah masa depan yang berharga.

2.4 Ringkasan

Dalam bab ini, kami telah membahas konsep LLM dan teknik terkait. Ini dapat dianggap sebagai pengantar umum, meskipun tidak menyeluruh, tentang LLM, yang menjadi dasar untuk diskusi lebih lanjut tentang topik yang lebih maju dalam bab-bab berikutnya. Lebih jauh, kami telah mengeksplorasi dua cara untuk meningkatkan LLM. Yang pertama berfokus pada pra-pelatihan LLM skala besar, yang sangat penting untuk mengembangkan model canggih. Yang kedua berfokus pada metode untuk mengadaptasi LLM ke input yang panjang, termasuk mengoptimalkan model perhatian, merancang cache KV yang lebih efisien dan terkompresi, menggabungkan model memori, dan mengeksplorasi penempatan posisi yang lebih baik.

Kekuatan LLM terletak pada kemampuannya untuk mendobrak batasan pelatihan model NLP untuk sejumlah tugas tertentu. Sebaliknya, LLM belajar dari sejumlah besar teks melalui tugas sederhana prediksi token — kami memprediksi token berikutnya dalam sebuah kalimat berdasarkan token sebelumnya.

¹⁸https://github.com/gkamradt/LLMTest_NeedleInAHaystack

Pandangan umum adalah bahwa, dengan mengulangi tugas prediksi token ini sejumlah besar kali, LLM dapat memperoleh pengetahuan tentang dunia dan bahasa, yang kemudian dapat diterapkan pada tugas-tugas baru. Sebagaimana Hasilnya, LLM dapat diminta untuk melakukan tugas apa pun dengan membingkainya sebagai tugas memprediksi tindakan selanjutnya. token yang diberikan perintah. Kemampuan yang muncul dalam model bahasa ini berasal dari beberapa dimensi, seperti peningkatan pelatihan, ukuran model, dan ukuran konteks. Tidak dapat disangkal bahwa hukum penskalaan saat ini merupakan prinsip dasar yang diadopsi dalam pengembangan model bahasa yang besar, meskipun hanya meningkatkan ukuran model belum terbukti cukup untuk mencapai AGI.

LLM telah ditemukan menunjukkan kemampuan dalam pemahaman, pembuatan, dan penalaran bahasa tujuan umum. Baru-baru ini, telah ditemukan bahwa peningkatan komputasi pada waktu inferensi juga dapat menyebabkan peningkatan signifikan dalam tugas penalaran yang kompleks [OpenAI, 2024].

Mengingat kekuatannya yang luar biasa, LLM telah menarik minat yang besar, baik dari segi teknik maupun aplikasinya. Akibatnya, ledakan minat penelitian terhadap LLM juga telah menyebabkan sejumlah besar teknik dan model baru. Namun, kami tidak berusaha memberikan tinjauan pustaka yang komprehensif tentang semua aspek LLM, mengingat evolusi bidang ini yang cepat. Namun demikian, seseorang masih bisa mendapatkan pengetahuan tentang LLM dari tinjauan umum [Zhao et al., 2023; Minaee et al., 2024] atau diskusi yang lebih terfokus pada topik tertentu [Ruan et al., 2024].

BAB 3

Dorongan

Dalam konteks LLM, prompting mengacu pada metode pemberian input atau isyarat tertentu kepada LLM untuk menghasilkan output yang diinginkan atau melakukan tugas. Misalnya, jika kita ingin LLM menerjemahkan kalimat dari bahasa Inggris ke bahasa Mandarin, kita dapat meminta seperti ini

Terjemahkan teks dari bahasa Inggris ke bahasa Mandarin.

Teks: Burung yang bangun pagi akan mendapat cacing.

Terjemahan: _____

Prompting sangat penting untuk LLM karena secara langsung memengaruhi seberapa efektif model ini memahami dan menanggapi pertanyaan pengguna. Prompt yang dibuat dengan baik dapat memandu LLM untuk menghasilkan respons yang lebih akurat, relevan, dan sesuai konteks. Lebih jauh, proses ini dapat disempurnakan secara berulang. Dengan menganalisis respons LLM, pengguna dapat menyesuaikan prompt mereka agar lebih sesuai dengan kebutuhan spesifik mereka. Mengingat pentingnya prompting dalam menerapkan LLM, desain prompt telah menjadi keterampilan penting bagi pengguna dan pengembang yang bekerja dengan LLM. Hal ini mengarah pada area penelitian aktif, yang disebut rekayasa prompt, di mana kami merancang prompt yang efektif untuk memanfaatkan LLM dengan lebih baik dan meningkatkan utilitas praktisnya dalam aplikasi dunia nyata.

Konsep penting yang terkait dengan prompting adalah pembelajaran dalam konteks. Saat mendorong LLM, kita dapat menambahkan informasi baru ke dalam konteks, seperti demonstrasi pemecahan masalah. Hal ini memungkinkan LLM untuk belajar dari konteks ini tentang cara memecahkan masalah. Berikut adalah contoh LLM yang mendorong dengan beberapa demonstrasi tentang cara mengklasifikasikan teks berdasarkan polaritas sentimen.

Berikut adalah beberapa contoh klasifikasi teks.

Contoh 1: Kami makan malam bersama dengan gembira. ý Label: Positif Contoh 2:

Saya frustrasi dengan penundaan ini. ý Label: Negatif Apa label untuk "Komentar itu cukup menyakitkan."?

Label: _____

Pembelajaran dalam konteks sering kali dilihat sebagai kemampuan baru dalam LLM yang muncul setelah pra-pelatihan. Meskipun LLM dapat dilatih atau disesuaikan untuk melakukan tugas-tugas baru, pembelajaran dalam konteks menyediakan cara yang sangat efisien untuk mengadaptasi model-model ini tanpa upaya pelatihan atau penyesuaian apa pun. Mungkin ini adalah salah satu fitur LLM yang paling menonjol: LLM memang mempelajari pengetahuan umum tentang dunia dan bahasa selama pra-pelatihan, yang dapat dengan mudah kita terapkan pada tantangan-tantangan baru. Selain itu, pembelajaran dalam konteks mencerminkan tren yang lebih luas untuk membuat sistem AI lebih dapat digeneralisasikan dan ramah pengguna. Alih-alih mengharuskan teknisi khusus menyempurnakan model untuk setiap tugas unik, pengguna dapat berinteraksi dengan LLM dengan cara yang lebih intuitif, cukup dengan memberikan contoh atau menyesuaikan konteks sebagaimana diperlukan.

Dalam bab ini, kami fokus pada teknik prompting dalam LLM. Kami mulai dengan mempertimbangkan beberapa desain prompt menarik yang umum digunakan dalam rekayasa prompt. Kemudian, kami membahas serangkaian

penyempurnaan pada metode ini. Terakhir, kami mengeksplorasi pendekatan untuk mengotomatiskan desain prompt.

3.1 Desain Prompt Umum

Bagian ini menyajikan konsep dasar dalam desain prompt, beserta contoh cara membuat prompt LLM untuk berbagai tugas NLP. Karena efektivitas prompt sangat bergantung pada LLM yang digunakan, prompt sering kali berbeda-beda di berbagai LLM, sehingga sulit untuk menyediakan daftar prompt yang komprehensif untuk semua LLM dan tugas-tugas selanjutnya. Oleh karena itu, pembahasan ini tidak difokuskan pada LLM tertentu. Sebaliknya, tujuannya adalah untuk menyediakan prinsip-prinsip panduan untuk desain prompt.

3.1.1 Dasar-dasar

Istilah prompt digunakan dalam berbagai cara. Dalam bab ini kami mendefinisikan prompt sebagai teks masukan ke LLM, dilambangkan dengan x . LLM menghasilkan teks y dengan memaksimalkan probabilitas $\Pr(y|x)$. Dalam proses pembuatan ini, perintah tersebut bertindak sebagai kondisi di mana kita membuat prediksi, dan dapat berisi informasi apa pun yang membantu menjelaskan dan memecahkan masalah.

Prompt dapat diperoleh dengan menggunakan template prompt (atau disingkat template) [Liu et al., 2023a]. Templat adalah bagian teks yang berisi placeholder atau variabel, yang setiap placeholder-nya dapat diisi dengan informasi tertentu. Berikut ini adalah dua templat untuk meminta saran akhir pekan dari LLM.

Tolong berikan saya beberapa saran untuk akhir pekan yang menyenangkan.

Jika {ypremisý}, apa saran Anda untuk akhir pekan yang menyenangkan?

Pada templat pertama, kita cukup menginstruksikan LLM untuk mengembalikan beberapa saran. Jadi templat tersebut hanyalah sepotong teks tanpa variabel. Pada templat kedua, variabel {ypremiseý} perlu ditentukan oleh pengguna untuk menyediakan premis dalam membuat saran. Misalnya, jika kita memasukkan

premis = cuacanya bagus akhir pekan ini

lalu kita bisa membuat prompt

Jika cuaca bagus akhir pekan ini, apa
sarana untuk akhir pekan yang menyenangkan?

Kita juga dapat mendesain template dengan beberapa variabel. Berikut adalah contoh di mana kita membandingkan dua kalimat tersebut berdasarkan kesamaan semantiknya.

Berikut ini adalah
 kalimat {ysentence1}
 Berikut kalimat lainnya
 {ykalimat2}
 Hitunglah kesamaan semantik antara dua kalimat tersebut

Cara populer untuk memformat prompt adalah dengan menulis setiap masukan atau keluaran dalam gaya “nama:konten”. Misalnya, kita dapat menggambarkan percakapan antara dua orang, bernama John dan David, dan menggunakan LLM untuk melanjutkan percakapan. Template untuk perintah tersebut diberikan oleh

Yohanes: {yucapan1}
 David: {yucapan2}
 Yohanes: {yucapan3}
 David: {yucapan4}
 Yohanes: {yucapan5}
 David: {yucapan6}
 Yohanes: {yucapan7}
 Daud: _____

Format “name:content” dapat digunakan untuk menentukan tugas yang ingin kita lakukan oleh LLM. Misalnya, karena “Q” dan “A” adalah singkatan yang umum digunakan untuk “Question” dan “Answer”, kita dapat menggunakan templat berikut untuk melakukan tanya jawab.

T: {ypertanyaan}
 A: _____

Format ini dapat digunakan untuk menjelaskan tugas yang lebih kompleks. Misalnya, berikut ini adalah contoh penyediaan spesifikasi untuk tugas penerjemahan

Tugas: Penerjemahan
Bahasa sumber: Bahasa Inggris
Bahasa target: Cina
Gaya: Teks formal
Template: Terjemahkan kalimat berikut: {ysentence}

Dalam sistem praktis, data tersebut biasanya direpresentasikan dan disimpan dalam pasangan kunci dan nilai, seperti Format JSON¹.

Ketika masalah sulit dijelaskan berdasarkan atribut, lebih umum untuk memberikan instruksi LLM dengan deskripsi yang jelas dan terperinci. Ada banyak cara untuk melakukan ini. Salah satunya

¹Representasi JSON adalah

Contohnya adalah menetapkan peran pada LLM dan menyediakan konteks yang memadai. Berikut ini adalah template yang menginstruksikan LLM untuk bertindak sebagai pakar dan menjawab pertanyaan dari anak-anak.

Anda adalah ilmuwan komputer dengan pengetahuan luas di bidang pembelajaran mendalam.

Tolong jelaskan konsep terkait komputer berikut kepada anak berusia sekitar 10 tahun, gunakan contoh sederhana jika memungkinkan. {ÿkonsepÿ}

Di sini teks "Anda adalah seorang ilmuwan komputer ... pembelajaran mendalam." terkadang disebut informasi sistem, dan disediakan untuk membantu LLM memahami konteks atau kendala tugas yang diminta untuk dilakukan.

3.1.2 Pembelajaran dalam konteks

Pembelajaran dapat terjadi selama inferensi. Pembelajaran dalam konteks adalah salah satu metode tersebut, di mana perintah melibatkan demonstrasi pemecahan masalah, dan LLM dapat belajar dari demonstrasi ini cara memecahkan masalah baru. Karena kami tidak memperbarui parameter model dalam proses ini, pembelajaran dalam konteks dapat dilihat sebagai cara untuk mengaktifkan dan mengatur ulang pengetahuan yang dipelajari dalam pra-pelatihan secara efisien tanpa pelatihan tambahan atau penyempurnaan. Hal ini memungkinkan adaptasi cepat LLM terhadap masalah baru, mendorong batasan dari apa yang dapat dicapai LLM yang telah dilatih sebelumnya tanpa penyesuaian khusus tugas.

Pembelajaran dalam konteks dapat diilustrasikan dengan membandingkan tiga metode: pembelajaran zero-shot, pembelajaran one-shot, dan pembelajaran few-shot. Pembelajaran zero-shot, seperti yang tersirat dalam namanya, tidak melibatkan proses "pembelajaran" tradisional. Sebaliknya, pembelajaran ini secara langsung menerapkan LLM untuk mengatasi masalah baru yang tidak diamati selama pelatihan. Dalam praktiknya, kita dapat menyesuaikan perintah secara berulang untuk memandu LLM dalam menghasilkan respons yang lebih baik, tanpa menunjukkan langkah-langkah pemecahan masalah a priori. Pertimbangkan contoh berikut. Misalkan kita ingin menggunakan LLM sebagai asisten yang dapat membantu mengoreksi kalimat bahasa Inggris. Prompt pembelajaran zero-shot diberikan oleh

```
{
  "Tugas": "Terjemahan"
  "Bahasa sumber": "Bahasa Inggris"
  "Bahasa target": "Cina"
  "Gaya": "Teks formal"
  "Template": "Terjemahkan kalimat berikut: {ÿsentenceÿ}"
}
```

SISTEM Anda adalah asisten yang membantu dan hebat dalam koreksi tata bahasa.

PENGGUNA Anda akan diberikan kalimat dalam bahasa Inggris. Tugasnya adalah untuk menghasilkan kalimat yang benar.

Masukan: Dia tidak suka pergi ke taman.

Keluaran: _____

Di sini kata-kata berwarna abu-abu digunakan untuk menunjukkan bidang perintah yang berbeda.

Dalam pembelajaran satu kali, kami memperluas perintah ini dengan menambahkan demonstrasi tentang cara memperbaiki kalimat, sehingga memungkinkan LLM untuk belajar dari pengalaman yang baru ditambahkan ini.

SISTEM Anda adalah asisten yang membantu dan hebat dalam koreksi tata bahasa.

DEMO Anda akan diberikan sebuah kalimat dalam bahasa Inggris. Tugasnya adalah menyusun kalimat yang benar.

Masukan: Ada banyak alasan untuk merayakannya.

Keluaran: Ada banyak alasan untuk merayakannya.

PENGGUNA Anda akan diberikan kalimat dalam bahasa Inggris. Tugasnya adalah untuk menghasilkan kalimat yang benar.

Masukan: Dia tidak suka pergi ke taman.

Keluaran: _____

Lebih jauh lagi, kita dapat menambahkan lebih banyak demonstrasi untuk memungkinkan pembelajaran dalam beberapa kali percobaan.

SISTEM Anda adalah asisten yang membantu dan hebat dalam koreksi tata bahasa.

DEMO1 Anda akan diberikan sebuah kalimat dalam bahasa Inggris. Tugasnya adalah menuliskan kalimat yang benar.

Masukan: Ada banyak alasan untuk merayakannya.

Keluaran: Ada banyak alasan untuk merayakannya.

DEMO2 Anda akan diberikan sebuah kalimat dalam bahasa Inggris. Tugasnya adalah menuliskan kalimat yang benar.

Masukan: Saya dan teman saya pergi ke pusat kebugaran setiap hari.

Keluaran: Saya dan teman saya pergi ke pusat kebugaran setiap hari.

PENGGUNA Anda akan diberikan kalimat dalam bahasa Inggris. Tugasnya adalah untuk menghasilkan kalimat yang benar.

Masukan: Dia tidak suka pergi ke taman.

Keluaran: _____

Dalam pembelajaran beberapa langkah, pada dasarnya kami menyediakan pola yang memetakan beberapa masukan ke keluaran yang sesuai. LLM mencoba mengikuti pola ini dalam membuat prediksi, asalkan perintah tersebut mencakup sejumlah demonstrasi yang cukup, meskipun umumnya kecil. Ini juga

mungkin untuk menggunakan pola yang lebih sederhana untuk mencapai hal ini. Misalnya, seseorang dapat menggunakan beberapa bidikan berikut petunjuk pembelajaran untuk menerjemahkan kata dari bahasa Mandarin ke bahasa Inggris.

DEMO $\text{ÿ} \text{ÿ} \text{ÿ}$ sekarang

$\text{ÿ} \text{ÿ} \text{ÿ}$ ayo

$\text{ÿ} \text{ÿ} \text{ÿ}$ pergi

$\text{ÿ} \text{ÿ} \text{ÿ}$ anak laki-laki

PENGGUNA $\text{ÿ} \text{ÿ} \text{ÿ}$

Jika LLM cukup kuat, pembelajaran dengan sedikit percobaan dapat memungkinkannya untuk mengatasi masalah-masalah yang rumit, seperti penalaran matematika. Misalnya, pertimbangkan tugas berikut untuk menjumlahkan dua angka lalu membagi hasilnya dengan hasil perkaliannya.

DEMO $12 \text{ } 5 \text{ } \text{ÿ} \text{ } (12 + 5)/(12 \times 5) = 0,283$

$3 \text{ } 1 \text{ } \text{ÿ} \text{ } (3 + 1)/(3 \times 1) = 1,33$

$-9 \text{ } 4 \text{ } \text{ÿ} \text{ } (\text{ÿ}9 + 4)/(\text{ÿ}9 \times 4) = 0,138$

$15 \text{ } 15 \text{ } \text{ÿ} \text{ } (15 + 15)/(15 \times 15) = 0,133$

PENGGUNA $19 \text{ } 73 \text{ } \text{ÿ}$

Dalam banyak aplikasi praktis, efektivitas pembelajaran dalam konteks sangat bergantung pada kualitas prompt dan kemampuan dasar LLM yang telah dilatih sebelumnya. Di satu sisi, kita membutuhkan upaya rekayasa cepat yang signifikan untuk mengembangkan permintaan yang tepat yang membantu LLM belajar lebih banyak efektif dari demonstrasi. Di sisi lain, LLM yang lebih kuat dapat memanfaatkan pembelajaran dalam konteks dengan lebih baik untuk melakukan tugas-tugas baru. Misalnya, anggaplah kita ingin menggunakan LLM untuk menerjemahkan kata dari Inuktitut ke Bahasa Inggris. Jika LLM tidak memiliki pelatihan awal tentang data Inuktitut, pemahaman tentang Inuktitut akan lemah, dan akan sulit bagi model untuk bekerja dengan baik terjemahan terlepas dari bagaimana kita memintanya. Dalam kasus ini, kita perlu melanjutkan pelatihan LLM dengan lebih banyak data Inuktitut, daripada mencoba mencari petunjuk yang lebih baik.

Mungkin menarik untuk mengeksplorasi bagaimana pembelajaran dalam konteks muncul selama pra-pelatihan dan mengapa hal ini berhasil selama inferensi. Satu pemahaman sederhana adalah bahwa LLM telah memperoleh beberapa pengetahuan tentang pemecahan masalah, tetapi ada banyak prediksi yang mungkin, yang sulit dibedakan ketika model menghadapi masalah baru. Memberikan demonstrasi dapat memandu LLM untuk mengikuti jalur yang “benar”. Lebih jauh, beberapa peneliti telah mencoba menafsirkan pembelajaran dalam konteks dari beberapa perspektif yang berbeda, termasuk inferensi Bayesian [Xie et al., 2022], gradien layak [Dai et al., 2023; Von Oswald et al., 2023], regresi linier [Akyürek et al., 2023], meta pembelajaran [Garg et al., 2022], dan seterusnya.

3.1.3 Strategi Rekayasa Cepat

Merancang prompt sangat empiris. Secara umum, ada banyak cara untuk mendorong LLM untuk melakukan tugas yang sama, dan kita perlu melakukan sejumlah uji coba dan kesalahan untuk menemukan prompt yang memuaskan. Untuk menulis prompt yang baik dengan lebih efisien, seseorang dapat mengikuti strategi tertentu. Contoh prinsip dorongan umum meliputi:

- Menjelaskan tugas sejelas mungkin. Saat kita menerapkan LLM untuk memecahkan masalah, kita perlu memberikan deskripsi masalah yang tepat, spesifik, dan jelas serta menginstruksikan LLM untuk melakukan seperti yang kita harapkan. Hal ini khususnya penting saat kita ingin hasil LLM memenuhi harapan tertentu. Misalnya, anggaplah kita ingin tahu tentang perubahan iklim. Perintah sederhana untuk meminta LLM memberikan beberapa informasi adalah

Ceritakan tentang perubahan iklim.

Karena instruksi ini terlalu umum, LLM dapat menghasilkan respons yang membahas aspek apa pun dari perubahan iklim, yang mungkin tidak sejalan dengan minat khusus kita. Dalam kasus ini, kita dapat menggunakan perintah yang spesifik dan terperinci. Salah satu contohnya adalah

Berikan penjelasan terperinci tentang penyebab dan dampak perubahan iklim, termasuk dampaknya terhadap suhu global, pola cuaca, dan permukaan laut. Selain itu, bahas solusi dan tindakan yang mungkin dilakukan untuk mengurangi dampak tersebut.

Sekarang, misalkan kita ingin menjelaskan perubahan iklim kepada anak berusia 10 tahun. Kita dapat menyesuaikan perintah di atas lebih lanjut.

Jelaskan penyebab dan dampak perubahan iklim kepada anak berusia 10 tahun.

Bicarakan tentang bagaimana hal itu memengaruhi cuaca, permukaan laut, dan suhu. Sebutkan juga beberapa hal yang dilakukan orang untuk membantu. Cobalah untuk menjelaskan dengan bahasa yang sederhana dan jangan melebihi 500 kata.

- Membimbing mahasiswa LLM untuk berpikir. Mahasiswa LLM telah menunjukkan kemampuan yang sangat baik untuk "berpikir". Contoh umum adalah bahwa LLM yang dikembangkan dengan baik telah mencapai kinerja yang mengesankan dalam tugas penalaran matematika, yang dianggap menantang. Dalam rekyasa cepat, kemampuan "berpikir" LLM perlu diaktifkan melalui dorongan yang tepat, terutama untuk masalah yang memerlukan upaya penalaran yang signifikan. Dalam banyak kasus, LLM yang diinstruksikan untuk "berpikir" dapat menghasilkan hasil yang sama sekali berbeda dibandingkan dengan LLM yang sama yang diinstruksikan untuk melakukan tugas secara langsung. Misalnya, [Kojima et al. \[2022\]](#) menemukan bahwa hanya dengan menambahkan "Mari berpikir selangkah demi selangkah" di akhir setiap dorongan dapat meningkatkan kinerja LLM pada beberapa tugas penalaran. LLM dapat diminta untuk "berpikir" dengan sejumlah cara. Salah satu metode adalah dengan menginstruksikan LLM untuk menghasilkan langkah-langkah untuk bernalar tentang masalah tersebut sebelum mencapai jawaban akhir. Misalnya, pertimbangkan tugas memecahkan masalah matematika. Lihat di bawah untuk dorongan sederhana untuk tugas ini.

**Anda seorang matematikawan. Anda akan diberikan soal matematika.
Tolong selesaikan masalahnya.**

Karena menyelesaikan soal matematika memerlukan proses penalaran yang terperinci, LLM mungkin akan membuat kesalahan jika mereka mencoba mencari jawabannya secara langsung. Jadi, kita dapat secara eksplisit meminta LLM untuk mengikuti proses penalaran tertentu sebelum mengambil kesimpulan.

Anda seorang matematikawan. Anda akan mengikuti langkah-langkah penalaran terperinci ini saat memecahkan soal matematika.

Langkah 1: Interpretasi Masalah.

Ahli matematika itu mendengarkan pertanyaan Anda dengan saksama dan memahami detail rumit tantangan matematika yang Anda sampaikan.

Langkah 2: Perumusan Strategi.

Dengan mengandalkan pengetahuannya yang luas, matematikawan memilih strategi paling efektif yang disesuaikan dengan jenis masalah matematika, apakah itu aljabar, kalkulus, atau geometri.

Langkah 3: Perhitungan Terperinci.

Dengan ketelitian dan keahlian, ahli matematika melakukan perhitungan yang diperlukan langkah demi langkah, mematuhi semua prinsip matematika.

Langkah 4: Tinjauan Solusi.

Sebelum memberikan jawaban akhir, ahli matematika dengan cermat memeriksa keakuratan perhitungan dan menawarkan penjelasan atau alasan ringkas untuk penyelesaiannya.

Anda akan diberikan soal matematika. Harap selesaikan soal tersebut.

{ŷmasalahŷ}

Metode lain untuk membimbing LLM agar “berpikir” adalah melalui beberapa putaran interaksi dengan LLM. Misalnya, sebagai langkah pertama, kita dapat menginstruksikan LLM untuk memecahkan masalah secara langsung

Anda akan diberikan soal matematika. Harap selesaikan soal tersebut. {ŷproblemŷ}

Sekarang kita memiliki jawaban awal untuk masalah tersebut. Sebagai langkah kedua, kami meminta LLM untuk mengevaluasi kebenaran jawaban tersebut dan, jika perlu, mengerjakannya ulang untuk menemukan solusi yang lebih baik.

Anda akan diberikan soal matematika beserta solusinya. Evaluasilah kebenaran solusi ini, dan identifikasi kesalahan jika ada. Kemudian, buatlah solusi Anda sendiri.

Masalah: {ŷmasalahŷ}

Solusi: {ŷsolusiŷ}

Pertanyaan yang disajikan di sini terkait erat dengan serangkaian penelitian panjang tentang masalah penalaran dalam LLM. Tidak mungkin untuk memberikan pembahasan lengkap tentang semua masalah terkait karena topik ini mencakup banyak metode. Namun, kita akan melihat pembahasan yang relatif lebih rinci tentang cara meningkatkan pertanyaan melalui penalaran yang lebih banyak di Bagian

- **Menyediakan informasi referensi.** Seperti yang dibahas di bagian sebelumnya, kita dapat menyertakan demonstrasi dalam prompt dan memungkinkan LLM untuk belajar dari demonstrasi ini tentang cara melakukan tugas. Faktanya, mengingat kemampuan pemahaman bahasa LLM yang luar biasa, kita dapat menambahkan jenis teks apa pun ke dalam prompt sehingga model ini dapat memprediksi berdasarkan konteks yang diperkaya. Dalam banyak aplikasi, kita memiliki berbagai informasi yang relevan dengan kueri pengguna. Alih-alih menggunakan LLM untuk membuat prediksi tanpa batasan, kita sering kali ingin LLM menghasilkan keluaran yang terbatas pada teks yang relevan. Salah satu contohnya adalah RAG, di mana teks yang relevan untuk kueri pengguna disediakan dengan memanggil sistem IR, dan kita meminta LLM untuk menghasilkan respons berdasarkan teks relevan yang disediakan ini. Prompt berikut menunjukkan sebuah contoh.

Anda adalah seorang pakar yang dapat menghasilkan jawaban atas pertanyaan yang dimasukkan. Anda kini telah diberikan pertanyaan dan informasi konteks yang sesuai. Harap buat jawaban berdasarkan informasi konteks ini. Perhatikan bahwa Anda perlu memberikan jawaban dengan kata-kata Anda sendiri, bukan sekadar menyalin dari konteks yang diberikan.

Informasi konteks: {ŷIR-resultŷ}

Kueri: {ŷqueryŷ}

Jika informasi konteksnya sangat dapat diandalkan, kita bahkan dapat membatasi LLM untuk menjawab hanya menggunakan teks yang disediakan. Contoh prompt ditunjukkan sebagai berikut

Anda adalah seorang ahli yang bertugas membuat jawaban dari kueri masukan. Anda telah diberikan kueri dan informasi konteks terkait, yang disusun dalam tabel, yang setiap barisnya mewakili rekaman yang berguna. Harap buat jawaban hanya dengan menggunakan informasi konteks ini. Pastikan Anda memberikan jawaban dengan kata-kata Anda sendiri.

Informasi konteks: {ütabely}

Kueri: {üqueryy}

Saat berhadapan dengan masalah di dunia nyata, kita sering kali memiliki pengetahuan sebelumnya dan informasi tambahan tentang masalah tersebut yang membantu menghasilkan jawaban yang lebih baik. Mempertimbangkan informasi tersebut dalam memberikan petunjuk umumnya membantu dalam meningkatkan hasil.

- Memperhatikan format prompt. Secara umum, kinerja LLM sangat sensitif terhadap prompt yang kita masukkan. Terkadang modifikasi kecil pada prompt dapat menyebabkan perubahan besar pada keluaran model. Contoh menariknya adalah bahwa mengubah urutan kalimat dalam prompt dapat menyebabkan LLM menghasilkan hasil yang berbeda. Untuk membuat prompt mudah dibaca dan mengurangi ambiguitas, biasanya formatnya dibuat sedemikian rupa sehingga memastikan kejelasan. Salah satu contohnya adalah kita mendefinisikan beberapa kolom untuk prompt dan mengisi informasi yang berbeda di setiap kolom. Contoh lainnya adalah kita dapat menggunakan prompt bergaya kode untuk LLM yang dapat memahami dan menghasilkan bahasa alami dan kode. Lihat yang berikut untuk prompt bergaya kode yang melakukan penerjemahan di mana satu demonstrasi disajikan.

[Bahasa Indonesia] = [Saya punya sebuah apel.]

[Jerman] = [Ich habe einen Apfel.]

[Bahasa Indonesia] = [Saya punya jeruk.]

[Jerman] = _____

LLM dapat menerima teks dalam berbagai format. Hal ini memungkinkan kita untuk menggunakan karakter kontrol, tag XML, dan format tertentu untuk merepresentasikan data yang kompleks. Dan berguna untuk menentukan bagaimana input dan output harus diformat atau disusun. Misalnya, kita dapat membatasi bagian teks menggunakan tanda kutip dan meminta LLM sesuai dengan itu (misalnya, menambahkan kalimat seperti "teks input dibatasi oleh tanda kutip ganda" ke permintaan).

Di atas, kami hanya membahas beberapa strategi untuk menulis prompt yang baik. Tentu saja, ada banyak metode seperti itu, dan seseorang perlu mengembangkan metodenya sendiri melalui praktik. Pembaca yang tertarik dapat merujuk ke berbagai dokumen daring untuk informasi lebih lanjut, seperti manual OpenAI tentang model seri GPT2

²Melihat <https://platform.openai.com/docs/guides/prompt-engineering/enam-strategi-untuk-mendapatkan-hasil-yang-lebih-baik>.

3.1.4 Contoh Lainnya

Dalam subbagian ini, kami akan mempertimbangkan lebih banyak contoh yang mendorong LLM untuk melakukan berbagai tugas NLP. Motivasinya di sini bukanlah untuk memberikan petunjuk standar untuk tugas-tugas ini, tetapi untuk menggunakan contoh-contoh sederhana guna menggambarkan bagaimana LLM dapat diminta untuk menangani masalah NLP.

3.1.4.1 Klasifikasi Teks

Klasifikasi teks mungkin merupakan salah satu masalah yang paling umum dalam NLP. Banyak tugas yang dapat dikategorikan secara luas sebagai pemberian label yang telah ditetapkan sebelumnya pada teks tertentu. Di sini kami mempertimbangkan masalah klasifikasi polaritas dalam analisis sentimen. Kami memilih klasifikasi polaritas sebagai ilustrasi karena ini merupakan salah satu tugas klasifikasi teks yang paling populer dan terdefinisi dengan baik. Dalam pengaturan umum klasifikasi polaritas, kami diminta untuk mengkategorikan teks tertentu ke dalam satu dari tiga kategori: negatif, positif, atau netral. Berikut adalah perintah sederhana untuk melakukan ini (agar mudah dibaca, kami menyorot deskripsi tugas dalam perintah tersebut).

Analisis polaritas teks berikut dan klasifikasikan sebagai positif, negatif, atau netral.

Teks:

Pelayanan di restoran lebih lambat dari yang diharapkan, yang agak membuat frustrasi.

Polaritas teks dapat diklasifikasikan sebagai positif.

Untuk melengkapi contoh, kami tampilkan respons yang dihasilkan oleh LLM (teks yang digarisbawahi).

Meskipun jawabannya benar, LLM memberikan jawaban ini bukan dalam label tetapi dalam teks yang menjelaskan hasilnya. Masalahnya adalah bahwa LLM dirancang untuk menghasilkan teks tetapi tidak untuk menetapkan label pada teks dan memperlakukan masalah klasifikasi sebagai masalah pembuatan teks. Akibatnya, kita memerlukan sistem lain untuk memetakan keluaran LLM ke ruang label (sebut saja pemetaan label), yaitu, kita mengekstrak "positif" dari "Polaritas teks dapat diklasifikasikan sebagai positif". Ini sepele dalam banyak kasus karena kita dapat mengidentifikasi kata-kata label melalui heuristik sederhana. Tetapi kadang-kadang, LLM mungkin tidak mengungkapkan hasil klasifikasi menggunakan kata-kata label ini. Dalam hal ini, masalahnya menjadi lebih rumit, karena kita memerlukan beberapa cara untuk memetakan teks atau kata-kata yang dihasilkan ke kata-kata label yang telah ditentukan sebelumnya.

Salah satu metode untuk menginduksi label keluaran dari LLM adalah dengan membungkai ulang masalah tersebut sebagai tugas cloze. Misalnya, berikut ini menunjukkan perintah seperti cloze untuk klasifikasi polaritas.

Analisis polaritas teks berikut dan klasifikasikan sebagai positif, negatif, atau netral.

Teks:

Pelayanan di restoran itu lebih lambat dari yang diharapkan, yang agak membuat frustrasi.

Polaritas teksnya positif

Kita dapat menggunakan LLM untuk melengkapi teks dan mengisi bagian yang kosong dengan kata yang paling tepat. Idealnya, kita ingin kata yang diisi adalah positif, negatif, atau netral. Namun, LLM tidak dijamin menghasilkan kata-kata label ini. Salah satu metode untuk mengatasi masalah ini adalah dengan membatasi prediksi terhadap kumpulan kata label dan pilih salah satu yang memiliki probabilitas tertinggi. Kemudian, output label diberikan oleh

$$\text{label} = \arg \max_{\substack{\text{Persamaan } (y) \\ x)} \text{ dan } (y|x)}} \quad (3.1)$$

di mana y menunjukkan kata yang diisi pada bagian yang kosong, dan Y menunjukkan kumpulan kata label {positif, negatif, netral}

Metode lain untuk menggunakan LLM untuk menghasilkan label adalah dengan membatasi output dengan perintah. Misalnya, kita dapat meminta LLM untuk memprediksi dalam serangkaian kata yang terkontrol. Berikut ini contohnya.

Analisislah polaritas teks berikut dan klasifikasikan sebagai positif, negatif, atau netral.

Teks:

Pelayanan di restoran itu lebih lambat dari yang diharapkan, yang agak membuat frustrasi.

Apa polaritas teksnya?

Jawab saja: positif, negatif, atau netral.

Positif

Analisis sentimen adalah masalah NLP umum yang mungkin sudah dipahami dengan baik oleh LLM melalui pra-pelatihan atau penyempurnaan. Dengan demikian, kita dapat mendorong LLM menggunakan instruksi sederhana untuk melakukan tugas. Namun, untuk masalah klasifikasi baru, mungkin perlu menyediakan rincian tambahan tentang tugas, seperti standar klasifikasi, sehingga LLM dapat melakukan dengan benar. Untuk melakukan ini, kita dapat menambahkan deskripsi tugas yang lebih rinci dan/atau mendemonstrasikan contoh klasifikasi dalam prompt. Sebagai ilustrasi, perhatikan contoh berikut.

Analisis polaritas teks berikut dan klasifikasikan sebagai positif, negatif, atau netral. Berikut ini adalah apa yang diwakili oleh setiap kategori:

Positif: Ini menunjukkan bahwa teks tersebut menyampaikan emosi atau sikap positif.

Misalnya, teks yang mengungkapkan kebahagiaan, kepuasan, kegembiraan, atau kekaguman dianggap positif.

Negatif: Ini merujuk pada teks yang mengekspresikan emosi atau sikap negatif. Meliputi perasaan sedih, marah, frustrasi, atau kritik.

Netral: Sentimen netral digunakan untuk menggambarkan teks yang tidak menunjukkan emosi positif atau negatif yang jelas, tetapi malah menyampaikan nada informasional, faktual, atau acuh tak acuh.

Teks:

Pelayanan di restoran lebih lambat dari yang diharapkan, yang agak membuat frustrasi.

Apa polaritas teksnya?

Positif

Meskipun tampaknya mudah untuk menggunakan LLM untuk masalah klasifikasi, masih ada masalah yang belum ditangani dengan baik. Misalnya, ketika berhadapan dengan sejumlah besar kategori, tetapi sulit untuk menjalankan LLM secara efektif. Perhatikan bahwa jika kita menghadapi masalah klasifikasi yang sangat sulit dan memiliki sejumlah data berlabel, penyempurnaan LLM atau penerapan arsitektur seperti "BERT + classifier" juga diinginkan.

3.1.4.2 Ekstraksi Informasi

Banyak masalah NLP dapat dianggap sebagai masalah ekstraksi informasi, yang melibatkan identifikasi atau ekstraksi bagian informasi tertentu dari teks yang tidak terstruktur. Informasi ini dapat mencakup entitas bernama, hubungan, peristiwa, dan titik data relevan lainnya. Tujuan ekstraksi informasi adalah untuk mengubah data mentah menjadi format yang dapat dengan mudah dianalisis dan digunakan dalam berbagai aplikasi hilir.

Karena ekstraksi informasi mencakup berbagai masalah, kami tidak dapat membahas semuanya di sini. Sebaliknya, kita mulai dengan tugas pengenalan entitas bernama — tugas yang telah lama menjadi perhatian dalam NLP. Pengenalan entitas bernama adalah proses yang mendekripsi dan mengklasifikasikan informasi penting dalam teks ke dalam kelompok-kelompok tertentu. Informasi penting ini, yang dikenal sebagai entitas bernama, biasanya mencakup nama diri dan dikategorikan ke dalam kelas-kelas yang berbeda seperti orang, lokasi, organisasi, tanggal, nilai moneter, dan persentase. Pertimbangkan terlebih dahulu contoh sederhana tentang mengekstraksi nama orang dari sebuah teks³

³Teksnya dari <https://edition.cnn.com/travel>

Identifikasi semua nama orang dalam teks yang disediakan.

Teks:

Apakah Inggris benar-benar melakukannya dengan buruk atau tren perjalanan telah berubah pascapandemi? Bagi Tom Jenkins, CEO Organisasi Pariwisata Eropa, yang terakhir adalah jawabannya. "Saya pikir Inggris berjalan dengan sangat baik, tetapi kita akan melihat lebih banyak orang pergi ke Eropa," katanya tentang tahun 2024, seraya menambahkan bahwa "diduga ada sedikit penurunan permintaan Amerika untuk Inggris."

Nama orang yang diidentifikasi dalam teks yang diberikan adalah: Tom Jenkins

Kemudian, kita dapat mengekstrak semua entitas bernama.

Identifikasi dan klasifikasikan semua entitas yang disebutkan dalam teks yang diberikan ke dalam kategori seperti nama orang, lokasi, tanggal, dan organisasi. Cantumkan setiap entitas beserta jenisnya dalam satu baris.

Teks:

Apakah Inggris benar-benar melakukannya dengan buruk atau tren perjalanan telah berubah pascapandemi? Bagi Tom Jenkins, CEO Organisasi Pariwisata Eropa, yang terakhir adalah jawabannya. "Saya pikir Inggris berjalan dengan sangat baik, tetapi kita akan melihat lebih banyak orang pergi ke Eropa," katanya tentang tahun 2024, seraya menambahkan bahwa "diduga ada sedikit penurunan permintaan Amerika untuk Inggris."

1. Tom Jenkins - Nama Orang

2. Organisasi Pariwisata Eropa - Organisasi

3. Inggris - Lokasi

4. Eropa - Lokasi

5. 2024 - Tanggal

Dengan entitas bernama yang teridentifikasi ini, kita dapat memproses teks lebih lanjut menggunakan tugas ekstraksi informasi lainnya. Misalnya, kita dapat mengidentifikasi hubungan antara entitas bernama ini (sebut saja ekstraksi relasi). Perintah terkait ditampilkan sebagai berikut.

Diberikan sebuah teks dan daftar entitas bernama yang diidentifikasi di dalamnya, analisis dan jelaskan hubungan antara entitas-entitas ini. Jelaskan bagaimana setiap entitas terkait secara kontekstual dengan entitas lainnya.

Teks:

Apakah Inggris benar-benar melakukannya dengan buruk atau tren perjalanan telah berubah pascapandemi? Bagi Tom Jenkins, CEO Organisasi Pariwisata Eropa, yang terakhir adalah jawabannya. "Saya pikir Inggris berjalan dengan sangat baik, tetapi kita akan melihat lebih banyak orang pergi ke Eropa," katanya tentang tahun 2024, seraya menambahkan bahwa "diduga ada sedikit penurunan permintaan Amerika untuk Inggris."

Entitas Bernama:

Tom Jenkins, Organisasi Pariwisata Eropa, Inggris, Eropa, 2024

Tom Jenkins - Organisasi Pariwisata Eropa: Tom Jenkins adalah CEO Organisasi Pariwisata Eropa. Ini menunjukkan adanya asosiasi profesional di mana Tom Jenkins kemungkinan memiliki peran penting dalam mengarahkan kegiatan organisasi, terutama yang terkait dengan pariwisata di Eropa.

...

Jika LLM telah disempurnakan dengan mengikuti instruksi untuk tugas ekstraksi informasi, relatif mudah untuk melakukan berbagai tugas ekstraksi informasi. Misalnya, berikut ini adalah templat perintah untuk ekstraksi informasi.

Anda akan diberikan teks. Tugas Anda adalah {*task-description*}

Teks: {*text*}

Di sini deskripsi tugas adalah instruksi untuk tugas tersebut. Contoh instruksinya meliputi:

mengekstrak kata kunci dari teks.

mengekstrak peristiwa utama dari teks.

mendeteksi hubungan koreferensi dan anaforis antara entitas dari teks.

dan seterusnya. Tentu saja, kami dapat memberikan deskripsi tugas yang lebih rinci untuk mendorong LLM dengan lebih akurat.

3.1.4.3 Pembuatan Teks

Dalam NLP, istilah text generation digunakan secara luas untuk mencakup berbagai macam masalah, yang banyak di antaranya berfokus pada pembuatan teks berdasarkan teks yang diberikan. Masalah-masalah ini secara garis besar dapat dikategorikan ke dalam dua kelas: 1) penulisan berkelanjutan berdasarkan teks masukan (sebut saja pelengkapan teks), dan 2) transformasi teks masukan menjadi teks lain (sebut saja transformasi teks). Di sini, kami mengilustrasikan petunjuk untuk yang pertama dan mempertimbangkan petunjuk untuk yang kedua

Contoh sederhana dari pelengkapan teks adalah melanjutkan menulis cerita dengan memulai beberapa kalimat yang diberikan. kalimat.

Anda seorang penulis yang hebat dalam menulis cerita. Anda diberikan beberapa kalimat untuk memulai. Silakan lanjutkan cerita dari kalimat yang diberikan.

Awal cerita: Pada

suatu sore yang cerah, Lily menemukan sebuah kunci emas kecil yang tersembunyi di bawah sebuah batu di halaman belakang rumahnya. Karena penasaran, ia mengambilnya dan bertanya-tanya apa yang mungkin bisa dibukanya. Tepat saat itu, kucingnya, Whiskers, mulai mengeong dengan gembira, menuntunnya ke arah pohon ek tua di tepi taman.

Lily mengikuti Whiskers, jantungnya berdebar kencang karena kegembiraan dan sedikit gugup. Pohon ek tua itu selalu tampak misterius baginya, cabang-cabangnya yang berbonggol-bonggol menghasilkan bayangan panjang bahkan pada hari-hari yang paling cerah. Saat mereka mencapai pohon itu, Whiskers melesat maju dan mulai meraba-raba sesuatu yang terkubur di bawah tumpukan daun musim gugur ...

Dengan cara yang sama, kita dapat membuat percakapan berdasarkan konteks percakapan.

Anda diberikan percakapan antara seorang turis dan seorang sopir taksi. Harap lengkapilah percakapan tersebut.

Turis: Bisakah Anda mengantar saya ke museum di pusat kota?

Sopir Taksi: Tentu saja! Baru pertama kali ke kota ini?

Turis: Ya, adakah tempat yang wajib dikunjungi di sekitar sini?

Sopir Taksi: Pastikan untuk mengunjungi kebun raya setelah museum.

Mereka dekat dan cukup cantik.

Turis: Bagus, terima kasih atas tipnya!

Sopir Taksi: Tidak masalah. Nikmati kunjungan Anda!

Aplikasi lain dari pelengkapan teks adalah pembuatan teks berdasarkan kebutuhan pengguna. Persyaratan tersebut dapat mencakup gaya, nada, panjang, dan konten spesifik apa pun yang harus ada dalam teks.

Misalnya, pengguna mungkin memerlukan laporan formal, cerita kreatif, atau potongan kode pemrograman.

Berikut ini adalah perintah untuk membuat puisi Mandarin teratur dengan lima aksara.

Anda seorang penyair. Silakan tulis puisi tradisional Tiongkok. Berikut persyaratannya.

Tema: Peremajaan musim semi

Struktur: Puisi yang diatur lima karakter

Emosi: Sampaikan rasa bahagia dan pembaruan melalui gambaran musim semi yang jelas.

yyyy

yyyy,yyyyyyy

yyyy,yyyyyy

Jika LLM dilatih untuk menghasilkan bahasa dan kode, kita dapat memintanya untuk melakukan pemrosesan kode. tugas penyelesaian. Berikut ini contohnya.

Silakan tulis fungsi Python untuk menghitung rata-rata dari daftar angka.

```
def calculate_average(angka): if
    numbers: # Periksa apakah daftar tersebut tidak
    kosong return sum(angka) / len(angka)
else:
    kembalikan 0 # Kembalikan 0 jika daftarnya kosong
```

3.1.4.4 Transformasi Teks

Tugas transformasi teks yang umum meliputi penerjemahan mesin, peringkasan, pemindahan gaya teks, dan sebagainya. Perintah yang digunakan untuk melakukan tugas-tugas ini tidak jauh berbeda dari yang disajikan sebelumnya. Di sini, kami menyajikan beberapa contoh untuk menunjukkan perintah-perintah ini.

Terjemahkan teks berikut dari bahasa Inggris ke bahasa Spanyol.

Teks:

Rubah coklat yang gesit melompati anjing malas.

Dengan cepat zorro marrón salta sobre el perro perezoso.

Rangkum artikel berikut ini dalam tidak lebih dari 50 kata:

Artikel:

Dalam beberapa tahun terakhir, kawasan perkotaan menghadapi tantangan signifikan terkait keberlanjutan dan pengelolaan lingkungan. Kota-kota mengalami tingkat yang lebih tinggi polusi, meningkatnya kemacetan lalu lintas, dan meningkatnya tuntutan terhadap infrastruktur. Pertumbuhan ini telah menyebabkan berbagai permasalahan lingkungan, termasuk meningkatnya polusi udara dan air, meningkatnya produksi sampah, dan beban masyarakat. layanan ...

Wilayah perkotaan tengah berjuang menghadapi tantangan keberlanjutan, seperti meningkatnya polusi, kemacetan lalu lintas, dan tuntutan infrastruktur
...

Tulis ulang teks ini dengan nada formal.

Teks:

Hai, apa kabar? Lama tak berjumpa!

Halo, apa kabar? Sudah lama sekali sejak terakhir kali kita bertemu!

3.1.4.5 Tanya Jawab

Format tanya jawab pada dasarnya sederhana. Untuk pertanyaan tertentu, ada jawaban yang sesuai dengannya. Misalnya, dalam menjawab pertanyaan domain terbuka, kami mengharapkan sistem untuk mengembalikan jawaban sebagai respons terhadap pertanyaan yang diajukan pengguna. Template prompt untuk keperluan umum menjawab pertanyaan dapat berupa:

{pertanyaan}

Pertanyaan: {pertanyaan}

Menjawab: _____

Menjawab pertanyaan penting dalam NLP karena banyak masalah dapat dibingkai sebagai tugas menjawab pertanyaan. Secara khusus, banyak tugas penalaran baru-baru ini didefinisikan dalam bentuk pertanyaan menjawab. Misalnya, dalam benchmark MMLU [Hendrycks et al., 2021], setiap contoh terdiri dari pertanyaan pilihan ganda, dan LLM diharuskan memilih jawaban yang benar. Lihat berikut adalah contoh petunjuk untuk menjawab pertanyaan dalam kumpulan data ini.

(Z, \circ) adalah grup dengan $a \circ b = a + b + 1$ untuk semua a, b di Z . Kebalikan dari a adalah

(Sebuah) 0

(B-2) adalah

(C) sebuah \circ 2

Contoh soal: $(2 + a) \circ \circ 1$

D

Tolok ukur lain yang banyak digunakan adalah kumpulan data GSM8K [Cobbe et al., 2021]. Kumpulan data ini terdiri dari ribuan soal cerita matematika sekolah dasar. Untuk setiap soal, kami menggunakan LLM untuk memberikan solusi dalam bahasa alami. Misalnya, berikut ini adalah perintah yang digunakan untuk menyelesaikan soal dalam GSM8K.

DEMO Q: Kedua putri Jerry bermain softball di tim yang berbeda. Mereka masing-masing bermain 8 kali musim ini. Setiap tim berlatih selama 4 jam untuk setiap pertandingan yang mereka ikuti. Jika setiap pertandingan berlangsung selama 2 jam, berapa jam Jerry akan menghabiskan waktu di lapangan untuk menonton kedua putrinya bermain dan berlatih bersama?

A: Jerry akan menghabiskan $8 \text{ pertandingan} * 2 \text{ jam per pertandingan} = 8 \circ 2 = 16 \circ 16 \text{ jam}$ untuk menonton salah satu putrinya bermain pertandingan. Ia akan menghabiskan $16 \circ 2 = 16 \circ 2 = 32 \circ 32 \text{ jam untuk menonton kedua putrinya bermain pertandingan. Ia akan menghabiskan } 8 \text{ pertandingan} * 4 \text{ jam latihan} = 8 \circ 4 = 32 \circ 32 \text{ jam untuk menonton salah satu putrinya berlatih. Ia akan menghabiskan } 32 \circ 2 = 32 \circ 2 = 64 \circ 64 \text{ jam untuk menonton kedua putrinya berlatih. Ia akan menghabiskan total } 32 \text{ jam untuk menonton pertandingan} + 64 \text{ jam untuk menonton latihan} = 32 + 64 = 96 \circ 96 \text{ jam. } \#\#\# 96$

T: Mary membeli enam buah apel dari toko. Dari apel yang dibelinya, untuk setiap apel yang dimakan Mary, ia menanam dua pohon dari apel yang tersisa.

Berapa banyak apel yang dimakan Maria?

A: Dia menanam delapan pohon. Ini berarti dia menggunakan setengah dari jumlah apel tersebut, yaitu $8 \text{ pohon} / 2 \text{ pohon / apel} = 8 / 2 = 4 \circ 4 \text{ apel}$. Itu berarti dia menanam empat dari enam apel yang dibelinya, sehingga hanya menyisakan $6 \text{ apel} - 4 \text{ apel} = 6 \circ 4 = 2 \circ 2 \text{ apel untuk dimakan. } \#\#\# 2$

PERTANYAAN PENGGUNA : Boris punya 100 apel. Beck punya 23 apel lebih sedikit daripada Boris.

Jika Boris memberi Beck 10 apel, berapa jumlah apel lebih sedikit yang dimiliki Beck daripada Boris sekarang?

A: Boris memulai dengan 100 apel dan memberikan Beck 10 apel, sehingga dia memiliki 100 apel $- 10 \text{ apel} = 100 - 10 = 90 \circ 90 \text{ apel}$.

Awalnya, Beck memiliki 23 apel lebih sedikit daripada Boris, sehingga jumlah awalnya adalah $100 - 23 = 100 - 23 = 77 \circ 77 \text{ apel}$. Setelah menerima 10 apel dari Boris, total Beck menjadi $77 \text{ apel} + 10 \text{ apel} = 77 + 10 = 87 \circ 87 \text{ apel}$. Akibatnya, Beck sekarang memiliki $90 - 87 = 90 - 87 = 3 \circ 3 \text{ apel lebih sedikit daripada Boris. } \#\#\# 3$

Di sini, prompt beberapa langkah diadopsi. LLM belajar dari demonstrasi pasangan masalah-solusi ini tidak hanya cara pemecahan masalah tetapi juga cara memformat output. Misalnya, hasil akhir perhitungan mengikuti token ####, dan $\text{y} \dots \text{y}$ memberi anotasi pada langkah-langkah perhitungan terperinci (disebut anotasi perhitungan)⁴.

3.2 Metode Prompting Lanjutan

Sejauh ini dalam bab ini, kita telah memperkenalkan konsep dasar yang terkait dengan prompting LLM dan menyajikan sejumlah prompt untuk tugas-tugas NLP. Sekarang kita akan mempertimbangkan beberapa teknik untuk meningkatkan efektivitas prompting.

3.2.1 Rantai Pikiran

Kita telah menjumpai konsep rantai pemikiran (CoT) beberapa kali dalam bab ini dan bab-bab sebelumnya [Wei et al., 2022c; Chowdhery et al., 2022]. Metode CoT menyediakan cara sederhana untuk mendorong LLM menghasilkan penalaran langkah demi langkah untuk masalah-masalah yang kompleks, sehingga mendekati tugas-tugas dengan cara yang lebih manusiawi. Daripada sampai pada kesimpulan secara langsung, metode CoT menginstruksikan LLM untuk menghasilkan langkah-langkah penalaran atau belajar dari demonstrasi proses penalaran terperinci yang disediakan dalam petunjuk. Untuk mengilustrasikan CoT, kita mempertimbangkan masalah kalkulasi aljabar, seperti yang umum dijelaskan dalam literatur. Misalkan kita diberi masalah aljabar

Hitunglah rata-rata angka 2, 4, dan 6.

Kita dapat menganggapnya sebagai pertanyaan dan meminta LLM untuk menjawabnya.

T: Tolong hitung rata-rata angka 2, 4, dan 9.

A: Jawabannya adalah 6.

Tampaknya sulit bagi LLM untuk memberikan jawaban yang benar secara langsung. Perbaikan sederhana adalah dengan menambahkan demonstrasi masalah serupa dalam soal, sehingga LLM dapat belajar dari demonstrasi tersebut.

T: Tolong hitung rata-rata angka 1, 3, 5, dan 7.

A: Jawabannya adalah 4.

T: Tolong hitung rata-rata angka 2, 4, dan 9.

A: Jawabannya adalah 7.

Masalahnya di sini adalah, meskipun kami telah menunjukkan pasangan pertanyaan-jawaban yang serupa, tetap sulit bagi LLM untuk menalar jawaban yang benar. Dalam CoT, LLM tidak hanya dapat belajar dari

⁴Selama prediksi, kalkulator digunakan saat kita melihat $\text{y} \dots \text{y}$. Lebih khusus lagi, setelah LLM menemukan “=” dalam $\text{y} \dots \text{y}$, maka kalkulator akan menghitung ekspresi di sisi kiri “=”. Metode ini membantu mengurangi kesalahan perhitungan yang dibuat oleh LLM.

korespondensi antara pertanyaan dan jawaban, tetapi mereka mungkin memperoleh lebih banyak manfaat dari langkah-langkah pemecahan masalah yang terperinci yang digunakan untuk memperoleh jawaban. Untuk melakukan ini, kita dapat menggabungkan beberapa langkah penalaran ke prompt untuk mendapatkan prompt CoT.

T: Tolong hitunglah kuadrat rata-rata dari angka 1, 3, 5, dan 7.

A: Hitunglah kuadrat setiap angka: $1^2 = 1$, $3^2 = 9$, $5^2 = 25$, dan $7^2 = 49$.

Jumlahkan kuadratnya, $1 + 9 + 25 + 49 = 84$. Totalnya ada 4 angka. Bagilah

jumlah dengan jumlah item, $84/4 = 21$. Jawabannya adalah 21.

T: Tolong hitung rata-rata angka 2, 4, dan 9.

A: Hitung $2 + 4 + 9$, yang sama dengan 15. Ada tiga angka. Bagilah

Jumlah total berdasarkan hitungan, menghasilkan $15/3 = 5$. Jawabannya adalah 5.

Di sini kami menyorot langkah-langkah penalaran dengan warna hijau. Dengan menyediakan proses penalaran yang terperinci untuk masalah serupa, LLM belajar untuk bernalar dan dengan demikian menghasilkan jalur pemecahan masalah yang mengarah pada jawaban yang benar.

Ada beberapa manfaat menggunakan prompting CoT. Pertama, CoT memungkinkan LLM untuk menguraikan masalah kompleks menjadi langkah-langkah penalaran berurutan yang lebih kecil. Hal ini agak mencerminkan manusia perilaku pemecahan masalah, membuatnya sangat efektif untuk tugas-tugas yang memerlukan langkah-langkah terperinci penalaran. Kedua, CoT membuat proses penalaran lebih transparan dan dapat ditafsirkan. Karena semua langkah-langkah penalaran terlihat, kita dapat memahami dan menafsirkan bagaimana suatu kesimpulan dicapai. Ketiga, jika pengguna dapat mengikuti logika di balik proses penalaran, mereka akan lebih cenderung mempercayai prediksi LLM. Hal ini khususnya penting ketika menerapkan LLM di bidang seperti kedokteran, pendidikan, dan keuangan. Keempat, CoT merupakan pendekatan pembelajaran dalam konteks, sehingga dapat diaplikasikan untuk sebagian besar LLM yang terlatih dan siap pakai. Selain itu, CoT menyediakan cara yang efisien untuk mengadaptasi LLM untuk berbagai jenis masalah. Bahkan dapat menginspirasi solusi yang lebih kreatif dengan mengeksplorasi berbagai jalur penalaran alternatif, yang mungkin tidak jelas saat mengambil kesimpulan secara langsung.

Metode yang dijelaskan di atas memerlukan penyediaan satu atau lebih contoh penalaran CoT, yang biasanya disebut metode CoT few-shot. Sebaliknya, metode CoT zero-shot tidak memerlukan contoh-contoh seperti itu. Sebaliknya, ia mendorong LLM untuk bernalar selangkah demi selangkah dengan memasukkan instruksi-instruksi khusus dalam prompt. Misalnya, di bawah ini adalah prompt CoT zero-shot.

T: Tolong hitung rata-rata angka 2, 4, dan 9.

A: Mari kita berpikir selangkah demi selangkah.

Kita memiliki tiga angka: 2, 4, dan 9. Tambahkan angka-angka ini bersama-sama,

$2 + 4 + 9 = 15$. Tentukan berapa banyak angka yang ada, yang dalam kasus ini

adalah tiga. Rata-rata dihitung dengan membagi jumlah total dengan jumlah

elemen. Jika pembagiannya dijumlahkan, hasilnya $15/3 = 5$. Jadi jawabannya adalah 5.

Mengikuti instruksi “Mari berpikir selangkah demi selangkah”, LLM diminta untuk menghasilkan langkah-langkah penalaran. Seperti yang dibahas dalam karya [Kojima et al. \[2022\]](#), mendorong dengan instruksi seperti itu dapat mengakibatkan LLM hanya menghasilkan langkah-langkah penalaran tanpa kesimpulan yang jelas. Dalam kasus ini, putaran kedua dari prompting dapat digunakan untuk mengekstrak jawaban dari langkah-langkah penalaran ini. Untuk misalnya, [Kojima et al. \[2022\]](#) membuat prompt kedua yang menggabungkan input dan output

pada putaran pertama dorongan. Dengan menggunakan masukan gabungan ini, LLM dapat melanjutkan proses penalarannya dan kemudian menghasilkan jawaban yang benar. Lebih jauh, LLM dapat diminta untuk bernalar menggunakan instruksi selain "Mari berpikir selangkah demi selangkah", seperti "Mari berpikir secara logis" dan "Tolong tunjukkan langkah-langkah berpikir Anda terlebih dahulu".

Meskipun kami telah mengilustrasikan metode CoT menggunakan masalah penalaran aljabar, metode ini dapat diterapkan pada berbagai masalah yang berbeda. Skenario pemecahan masalah yang umum untuk CoT meliputi penalaran matematika, penalaran logis, penalaran akal sehat, penalaran simbolik, pembuatan kode, dan sebagainya. Lihat Gambar 3.1 untuk contoh penerapan CoT dalam berbagai tugas.

CoT saat ini merupakan salah satu bidang rekayasa prompt yang paling aktif. Hal ini tidak hanya menghasilkan peningkatan kinerja prompting LLM, tetapi juga membuka pintu bagi berbagai metode untuk mempelajari dan memverifikasi kemampuan penalaran LLM. Meskipun kami telah berfokus pada ide dasar CoT di bagian ini, CoT dapat ditingkatkan dengan beberapa cara. Misalnya, kita dapat mempertimbangkan proses penalaran sebagai masalah pencarian melalui banyak kemungkinan jalur, yang masing-masing dapat terdiri dari beberapa status perantara (yaitu, langkah-langkah penalaran). Secara umum, kami ingin ruang pencarian terdefinisi dengan baik dan cukup besar, sehingga kami lebih mungkin menemukan hasil yang optimal. Karena alasan ini, area penelitian LLM saat ini ditujukan untuk merancang struktur yang lebih baik untuk merepresentasikan proses penalaran, yang memungkinkan LLM untuk mengatasi tantangan penalaran yang lebih kompleks. Struktur ini mencakup struktur berbasis pohon [Yao et al., 2024], struktur berbasis grafik [Besta et al., 2024], dan seterusnya. Dengan menggunakan representasi ringkas dari jalur penalaran ini, LLM dapat mengeksplorasi rentang jalur pengambilan keputusan yang lebih luas, yang Penelitian lain yang dilakukan analog dengan pemikiran Sistem 2⁵ cuses pada LLM yang mendorong dengan interaksi multi-putaran. Ini melibatkan penguraian masalah kompleks menjadi sub-masalah, verifikasi dan penyempurnaan keluaran model, penggunaan ensembling model, dan seterusnya. Perhatikan bahwa metode dan isu yang terlibat ini tidak terbatas pada CoT. Faktanya, mereka sering digunakan sebagai pendekatan yang lebih umum untuk meningkatkan LLM, sementara CoT dapat dilihat sebagai cara untuk menguji kemampuan LLM. Kita akan melihat pembahasan beberapa isu ini di subbagian berikut.

Sebelum mengakhiri pembahasan kita tentang CoT, kita harus mempertimbangkan keterbatasan praktisnya. Salah satunya adalah perlunya demonstrasi penalaran multi-langkah yang terperinci dalam skenario CoT few-shot, yang mungkin sulit diperoleh, baik secara otomatis maupun manual. Selain itu, tidak ada metode standar untuk memecah masalah kompleks menjadi langkah-langkah pemecahan masalah yang lebih sederhana. Hal ini sering kali sangat bergantung pada pengalaman pengguna. Selain itu, kesalahan dalam langkah-langkah perantara juga dapat memengaruhi keakuratan kesimpulan akhir. Untuk pembahasan lebih lanjut tentang pro dan kontra CoT, pembaca yang tertarik dapat merujuk ke survei terbaru tentang topik ini [Chu et al., 2023; Yu et al., 2023; Zhang et al., 2023].

3.2.2 Dekomposisi Masalah

Kita telah melihat bahwa LLM dapat memperoleh manfaat dari pemecahan masalah yang kompleks dengan memecahnya menjadi tugas pemecahan masalah yang lebih sederhana. Pendekatan semacam itu dapat dilihat sebagai contoh paradigma yang lebih luas yang dikenal sebagai dekomposisi masalah, yang telah dieksplorasi dan dibahas secara luas dalam psikologi dan ilmu komputer. Dari perspektif psikologis, pemecahan masalah yang kompleks mengacu pada proses mengatasi masalah menggunakan pengetahuan yang membantu mengatasi hambatan

⁵ Pemikiran Sistem 1 dan Sistem 2, sebagaimana dijelaskan oleh Kahneman [2011], merupakan dua mode pemrosesan kognitif yang berbeda. Sistem 1 cepat, otomatis, intuitif, dan emosional. Mode berpikir ini bekerja dengan mudah dan cepat, dan sering kali menjadi panduan keputusan, penilaian, dan kesan kita sehari-hari. Sistem 2 lambat, disengaja, dan analitis. Sistem ini diaktifkan saat kita perlu melakukan perhitungan yang rumit.

T: Peralatan hiburan rumah apa yang memerlukan kabel?

Pilihan Jawaban: (a) rumah radio (b) gardu induk (c) televisi (d) lemari

A: Jawabannya harus memerlukan kabel.
pilihan di atas, hanya televisi yang memerlukan kabel.
Jadi jawabannya adalah (c).

T: Tipe orang seperti apa yang biasanya tertular penyakit?

Pilihan Jawaban: (a) rumah sakit (b) kepala (c) orang sakit (d) orang tua (e) kantor dokter

A: Jawabannya haruslah orang yang rentan terhadap penyakit. Dari pilihan tersebut, "lansia"
"anak laki-laki" benar karena mereka biasanya memiliki sistem kekebalan tubuh yang lebih lemah.
Jadi jawabannya adalah (d).

T: Apakah hamster menyediakan makanan untuk hewan apa pun?

A: Hamster adalah hewan mangsa. Mangsa adalah makanan untuk predator. Jadi, hamster menyediakan makanan untuk beberapa hewan. Jadi jawabannya adalah ya.

T: Bisakah Anda menggunakan Microsoft Office tanpa internet?

A: Ya, Anda dapat menggunakan aplikasi Microsoft Office.
kation, meskipun fitur seperti penyimpanan
awan dan kolaborasi daring memerlukan
internet. Jadi jawabannya adalah ya.

(a) CSQA

(b) Strategi QA

T: Lengkapi sisa rangkaiannya, pastikan tanda kurung ditutup dengan benar. Masukan: [{

A: Mari kita berpikir selangkah demi selangkah.
0: tumpukan kosong

1: [; tumpukan: [
2: { ; tumpukan: [{
Jadi jawabannya adalah }].

T: Lengkapi sisa rangkaian, pastikan tanda kurung ditutup dengan benar. Masukan: < [[

A: Mari kita berpikir selangkah demi selangkah.
0: tumpukan kosong
1: < ; tumpukan: <
2: [; tumpukan: < [
3: [; tumpukan: < [[
Jadi jawabannya adalah]] >.

T: Ambil huruf terakhir dari kata-kata "Elon" Musk" dan menggabungkannya.

A: Huruf terakhir dari "Elon" adalah "n". Huruf terakhir huruf "Musk" adalah "k". Menggabungkannya adalah nk". Jawabannya adalah nk.

T: Ambil huruf terakhir dari setiap kata dalam "Renata Mariela Mona Kristin" dan gabungkan mereka.

A: Huruf terakhir dari "Renata" adalah "a".
huruf terakhir dari "Mariela" adalah "a". Huruf terakhir huruf terakhir dari "Mona" adalah "a". Huruf terakhir dari "Kristin" adalah "n". Menggabungkan keduanya adalah "aan". Jawabannya adalah aan.

(c) Bahasa Dyck

(d) Penggabungan Huruf Terakhir

Gbr. 3.1: CoT dalam empat tugas penalaran yang berbeda, termasuk CSQA, StrategyQA, bahasa Dyck, dan Last Letter Con-catenation. Bagian CoT disorot dengan warna hijau.

masalah6 . Umumnya tidak ada cara yang baku atau jelas untuk menemukan solusi atas masalah yang kompleks. Namun, seringkali lebih baik untuk menggunakan strategi yang menguraikan masalah tersebut, sehingga membuatnya lebih mudah untuk mengatasi sub-masalah yang sesuai dengan usaha yang lebih sedikit. Misalnya, pertimbangkan menulis blog tentang risiko AI. Jika kita hanya meminta LLM dengan instruksi “Silakan menulis blog tentang risiko AI”, LLM dapat menghasilkan blog dengan struktur yang sewenang-wenang dan

⁶Definisi yang relatif formal dapat ditemukan dalam buku Frensch dan Funke [2014] : pemecahan masalah yang kompleks terjadi untuk mengatasi hambatan antara keadaan tertentu dan keadaan tujuan yang diinginkan melalui perilaku dan/atau kognitif, multi-langkah kegiatan.

gaya penulisan. Metode yang lebih baik adalah dengan menguraikan blog dan memberikan informasi yang lebih rinci tentang setiap bagian. Pertimbangkan perintah berikut

Anda seorang penulis blog. Harap ikuti garis besar yang disediakan di bawah ini untuk menulis blog tentang risiko AI.

- Perkenalan

Perkenalkan AI, relevansinya, dan pentingnya memahami risikonya bagi kaum muda.

- Kekhawatiran Privasi

Diskusikan bagaimana AI dapat membahayakan privasi pribadi melalui interaksi daring.

- Misinformasi

Jelajahi peran AI dalam menyebarkan misinformasi dan memengaruhi keputusan kaum muda.

- Perundungan dunia maya

Menyorot bagaimana alat AI dapat digunakan dalam perundungan siber dan dampaknya terhadap kesehatan mental.

- Tips Penggunaan AI yang Aman

Menawarkan pedoman untuk penggunaan AI yang bertanggung jawab dan mempromosikan pemikiran kritis.

- Kesimpulan

Rangkum poin-poin utama dan dorong keterlibatan proaktif dengan etika AI.

Di sini kami memberikan judul dan poin utama untuk setiap bagian. Kemudian, LLM dapat menggunakan struktur ini untuk memecah tugas menulis dengan mengisi konten untuk bagian-bagian ini. Perhatikan bahwa cara menyusun blog dapat disediakan oleh manusia atau bahkan dibuat secara otomatis. Misalnya, kita dapat menggunakan LLM untuk terlebih dahulu membuat kerangka, lalu memintanya untuk mengikuti kerangka ini untuk menyelesaikan tulisan.

Dalam ilmu komputer, menguraikan masalah yang kompleks merupakan strategi yang umum digunakan dalam desain sistem perangkat lunak dan perangkat keras. Contoh yang terkenal adalah paradigma bagi-dan-kuasai, yang sering digunakan untuk merancang algoritma untuk masalah komputasi yang dapat disederhanakan menjadi masalah yang lebih mudah dikelola. Misalnya, pertimbangkan masalah untuk menentukan apakah suatu dokumen membahas risiko AI. Kita dapat menginstruksikan LLM dengan perintah berikut.

Anda diberikan sebuah teks. Harap tentukan apakah teks tersebut membahas risiko AI.

{`ydokumeny`}

Jika dokumennya panjang, perhitungannya akan mahal. Atau, kita dapat membagi dokumen menjadi segmen yang relatif pendek dan melakukan tugas yang sama pada setiap segmen. Segmen ini dapat diproses secara paralel untuk lebih mengurangi biaya perhitungan. Selanjutnya, kita menentukan

relevansi setiap segmen dengan topik risiko AI. Hasil akhir kemudian dihasilkan menggunakan perintah lain.

Tugas Anda adalah menentukan apakah teks tersebut membahas risiko AI. Teks ini memiliki telah dibagi menjadi beberapa segmen, dan Anda telah memperoleh relevansi setiap segmen dengan topik risiko AI. Berdasarkan hal ini, berikan hasil akhir Anda.

Segmen 1: {ŷrelevansi-dengan-topik1ŷ}

Segmen 2: {ŷrelevansi-dengan-topik2ŷ}

Segmen 3: {ŷrelevansi-dengan-topik3ŷ}

...

Sekarang mari kita kembali ke pembahasan yang lebih umum tentang dekomposisi masalah dalam prompting. Sementara dekomposisi masalah dapat diterapkan pada berbagai masalah NLP, hal itu telah dibahas dan diuji secara lebih luas dalam tugas-tugas penalaran baru-baru ini. Untuk tugas-tugas penalaran yang kompleks, kita sering membutuhkan jalur penalaran multi-langkah untuk mencapai kesimpulan yang benar. Kita dapat menggunakan LLM untuk mencapai ini dengan tiga cara berbeda. Pertama, LLM dapat langsung mencapai kesimpulan. Dengan kata lain, mereka dapat memprediksi tanpa proses penalaran eksplisit, dan ada mekanisme penalaran yang tersembunyi dan tidak dapat ditafsirkan. Kedua, LLM diminta untuk menghasilkan jalur penalaran multi-langkah yang mengarah ke kesimpulan, seperti CoT. Namun, kita menjalankan LLM hanya sekali, dan semua langkah perantara dalam penalaran dihasilkan dalam satu prediksi. Ketiga, kita memecah masalah asli menjadi sejumlah sub-masalah, yang ditangani dalam menjalankan LLM yang terpisah atau ditangani bersama-sama. Di sini kami memfokuskan perhatian pada pendekatan ketiga, yang terkait erat dengan dekomposisi masalah. Namun, perlu dicatat bahwa pembahasan yang lebih komprehensif dapat mencakup semua pendekatan ini, sedangkan dua pendekatan pertama telah dibahas sampai batas tertentu dalam bab ini.

Kerangka kerja umum untuk dekomposisi masalah melibatkan dua elemen.

- **Pembuatan Sub-Permasalahan.** Ini melibatkan penguraian masalah masukan menjadi sejumlah sub-masalah.
- **Pemecahan Sub-masalah.** Ini melibatkan pemecahan setiap sub-masalah dan memperoleh kesimpulan antara dan akhir melalui penalaran.

Kedua masalah ini dapat dimodelkan dengan cara yang berbeda, yang mengarah ke berbagai metode dekomposisi masalah. Salah satu pendekatan adalah memperlakukannya sebagai langkah terpisah dalam proses dua langkah. Misalnya, pertimbangkan tugas menulis blog yang dijelaskan di awal subbagian ini. Pada langkah pertama, kami menguraikan seluruh masalah menjadi sub-masalah sekaligus (yaitu, menguraikan blog). Pada langkah kedua, kami memecahkan sub-masalah secara berurutan atau dalam urutan lain (yaitu, mengisi konten untuk setiap bagian sesuai kebutuhan). Output akhir dari proses ini menggabungkan hasil dari penyelesaian setiap sub-masalah. Meskipun metode ini sederhana dan mudah, metode ini mengasumsikan bahwa masalahnya bersifat komposisi, sehingga lebih cocok untuk tugas-tugas seperti penulisan dan pembuatan kode.

Namun, banyak masalah di dunia nyata memerlukan penalaran yang kompleks. Salah satu karakteristik utama dari masalah ini adalah bahwa langkah-langkah penalaran mungkin tidak tetap. Alur penalaran dapat bervariasi untuk masalah yang berbeda, dan setiap langkah penalaran mungkin bergantung pada hasil dari langkah-langkah sebelumnya.

Dalam kasus seperti itu, tidaklah diinginkan untuk menggunakan pembuatan sub-masalah tetap terlebih dahulu. Sebaliknya, sub-masalah harus dibuat secara dinamis berdasarkan masalah masukan, dan, jika memungkinkan, dibuat dengan cepat selama proses penalaran. Hal ini membuat dekomposisi masalah lebih menantang dibandingkan dengan merancang algoritma bagi-dan-kuasai. Idealnya, kami ingin merancang kedua sistem untuk pembuatan sub-masalah dan penyelesaian sub-masalah secara bersamaan. Namun, pendekatan yang lebih praktis dan banyak digunakan adalah dengan mengadopsi model terpisah untuk tugas-tugas ini. Cara mudah untuk mencapainya adalah dengan mengadaptasi LLM untuk tugas-tugas ini dengan meminta atau menyetel model.

Di sini kami mempertimbangkan metode berdasarkan ide di atas, yang disebut least-to-most prompting [Zhou et al., 2023b]. Motivasi untuk metode ini muncul dari tantangan dalam memecahkan masalah penalaran yang sulit — masalah yang tidak dapat diatasi hanya dengan menggeneralisasi dari beberapa contoh. Untuk masalah ini, strategi pemecahan masalah yang lebih efektif adalah mengikuti urutan submasalah progresif yang secara sistematis mengarah pada kesimpulan. Lebih khusus lagi, dalam metode least-to-most prompting, pembuatan submasalah dilakukan dengan meminta LLM dengan instruksi dan/atau demonstrasi. Misalnya, di bawah ini adalah prompt 2-shot untuk pembuatan submasalah dalam least-to-most prompting.

TUGAS Tugas Anda adalah menguraikan suatu masalah menjadi beberapa sub-masalah. Anda akan diberikan beberapa contoh untuk mengilustrasikan cara mencapainya.

DEMO Q: Di suatu komunitas, 5% dari populasi adalah bayi, 15% adalah anak-anak, 40% adalah orang dewasa, dan 40% adalah orang tua. Kelompok mana yang merupakan bagian terbesar dari populasi?

A: Untuk menjawab pertanyaan “Kelompok manakah yang merupakan bagian terbesar dari populasi?”, kita perlu mengetahui: “Berapa persen yang merupakan bayi?”, “Berapa persen yang merupakan anak-anak?”, “Berapa persen yang merupakan orang dewasa?”, “Berapa persen yang merupakan orang tua?”.

T: Alice, Bob, dan Charlie membawa manik-manik untuk proyek kelompok mereka di kelas kerajinan. Alice memiliki manik-manik dua kali lebih banyak daripada Bob, dan Bob memiliki manik-manik lima kali lebih banyak daripada Charlie. Jika Charlie memiliki 6 manik-manik, berapa banyak manik-manik yang dapat mereka gunakan untuk proyek kerajinan mereka?

A: Untuk menjawab pertanyaan “Berapa banyak manik-manik yang dapat mereka gunakan untuk proyek kerajinan mereka?”, kita perlu mengetahui: “Berapa banyak manik-manik yang dimiliki Bob?”, “Berapa banyak manik-manik yang dimiliki Alice?”.

PERTANYAAN PENGGUNA : Studi lingkungan yang dilakukan dari tahun 2015 hingga 2020 mengungkapkan bahwa suhu rata-rata di wilayah tersebut meningkat sebesar 2,3 derajat Celsius. Berapa lama durasi studi lingkungan tersebut?

A: Untuk menjawab pertanyaan “Berapa lama durasi studi lingkungan?”, kita perlu mengetahui: “Kapan studi lingkungan dimulai?”, “Kapan studi lingkungan berakhir?”.

Dengan mempelajari contoh-contoh tersebut, LLM dapat menghasilkan dua sub-masalah untuk menjawab masalah baru “Berapa lama durasi studi lingkungan?” (disorot dengan warna biru dan oranye). Dengan sub-masalah ini, kami menyelesaikannya secara berurutan. Untuk setiap sub-masalah, kami mengambil semua pasangan QA yang dihasilkan sebelumnya sebagai konteks, lalu menghasilkan jawabannya. Untuk contoh di atas,

kita perlu menjawab sub-masalah pertama dengan meminta LLM, seperti ini

Studi lingkungan yang dilakukan dari tahun 2015 hingga 2020 mengungkapkan bahwa suhu rata-rata di wilayah tersebut meningkat sebesar 2,3 derajat Celsius.

SUB-PROB1 T: Kapan studi lingkungan dimulai?

A: Studi lingkungan dimulai pada tahun 2015.

Setelah kita mendapatkan jawaban untuk sub-masalah pertama, kita lanjut ke sub-masalah kedua. Kali ini, kami menyertakan sub-masalah pertama dan jawabannya dalam masukan.

Studi lingkungan yang dilakukan dari tahun 2015 hingga 2020 mengungkapkan bahwa suhu rata-rata di wilayah tersebut meningkat sebesar 2,3 derajat Celsius.

SUB-PROB1 T: Kapan studi lingkungan dimulai?

A: Studi lingkungan dimulai pada tahun 2015.

SUB-PROB2 T: Kapan studi lingkungan berakhir?

A: Studi lingkungan dimulai pada tahun 2020.

Terakhir, kami menggunakan LLM untuk memecahkan masalah asli dengan jawaban untuk semua sub-masalah.

Studi lingkungan yang dilakukan dari tahun 2015 hingga 2020 mengungkapkan bahwa suhu rata-rata di wilayah tersebut meningkat sebesar 2,3 derajat Celsius.

SUB-PROB1 T: Kapan studi lingkungan dimulai?

A: Studi lingkungan dimulai pada tahun 2015.

SUB-PROB2 T: Kapan studi lingkungan berakhir?

A: Studi lingkungan dimulai pada tahun 2020.

Pertanyaan AKHIR : Berapa lama durasi studi lingkungan?

A: Durasi studi lingkungan adalah 5 tahun.

Metode least-to-most menawarkan pendekatan dasar untuk mendorong LLM untuk menghasilkan dan memecahkan sub-masalah secara terpisah. Kita dapat meningkatkannya dengan beberapa cara. Salah satu peningkatan sederhana adalah dengan menerapkan berbagai teknik dorongan tingkat lanjut, yang tidak memerlukan perubahan pada kerangka dekomposisi masalah. Misalnya, kita dapat memasukkan CoT ke dalam dorongan untuk meningkatkan kinerja penalaran dalam pembuatan dan penyelesaian sub-masalah.

Peningkatan lainnya adalah mengeksplorasi metode untuk menguraikan masalah dengan lebih baik dan mengatur jalur pemecahan masalah. Untuk menggambarkan pendekatan ini, kita akan menggunakan simbol p0 untuk menunjukkan

masalah masukan, dan menggunakan simbol $\{p_1, \dots, p_n\}$ untuk menunjukkan sub-masalah yang sesuai dengan p_0 .

Untuk dorongan paling sedikit ke paling banyak, kita menguraikan p_0 menjadi $\{p_1, \dots, p_n\}$, yang diberikan oleh

$$\{p_1, \dots, p_n\} = G(p_0) \quad (3.2)$$

di mana $G(\cdot)$ menyatakan fungsi pembangkitan sub-masalah. Kemudian, kita selesaikan sub-masalah $\{p_1, \dots, p_n\}$ secara berurutan, yang menghasilkan urutan jawaban $\{a_1, \dots, a_n\}$. Untuk menjawab sub-masalah ke- i p_i , kita sertakan masalah asli p_0 dan semua pasangan masalah-jawaban yang terlihat sebelumnya dalam konteks untuk prediksi. Jawaban a_i diberikan oleh

$$a_i = Si(p_i, \{p_0, p_{<i}, a_{<i}\}) \quad (3.3)$$

di mana $p_{<i} = \{p_1, \dots, p_{i-1}\}$ dan $a_{<i} = \{a_1, \dots, a_{i-1}\}$. $Si(\cdot)$ menyatakan fungsi yang menyelesaikan sub-masalah p_i dengan konteks $\{p_0, p_{<i}, a_{<i}\}$. Langkah terakhir adalah menghasilkan jawaban untuk masalah awal p_0 , yang dapat dinyatakan dengan cara yang mirip dengan Persamaan (3.3).

$$a_0 = So(p_0, p_{<n}, a_{<n}) \quad (3.4)$$

Salah satu cara untuk menyempurnakan model ini adalah dengan memodifikasi fungsi $G(\cdot)$ sehingga model dapat menghasilkan jawaban secara dinamis. Daripada menghasilkan semua sub-masalah pada satu waktu, kita dapat menghasilkan masing-masing dari mereka selama pemecahan masalah [Dua et al., 2022]. Untuk melakukan ini, kita dapat mengganti Persamaan (3.2) dengan

$$p_i = Gi(p_0, \{p_{<i}, a_{<i}\}) \quad (3.5)$$

Oleh karena itu, kami memperoleh model pembangkitan sub-masalah yang beroperasi secara bertahap. Pada setiap langkah i , pertama-tama kami membangkitkan sub-masalah p_i dengan memunculkan LLM dengan masalah awal p_0 dan riwayat pemecahan masalah $\{p_{<i}, a_{<i}\}$. Kemudian kami membangkitkan jawaban a_i untuk sub-masalah ini menggunakan LLM yang sama atau berbeda, berdasarkan informasi kontekstual yang sama (lihat Persamaan (3.3)). Metode ini secara efektif memperluas kapasitas penalaran LLM dengan memungkinkan mereka membangkitkan dan memecahkan sub-masalah secara dinamis dalam langkah-langkah penalaran menengah. Akibatnya, jalur penalaran tidak ditetapkan sebelumnya, dan model dapat memilih dan mengadaptasi strategi penalaran mereka selama pemecahan masalah.

Cara lain untuk meningkatkan model di atas adalah dengan berfokus pada pengembangan pemecah sub-masalah yang lebih baik. Dalam pembahasan kita sebelumnya, kita membatasi $Si(\cdot)$ pada LLM yang diminta untuk memecahkan sub-masalah. Faktanya, kita dapat memperluas fungsi ini ke sistem apa pun yang mampu menangani sub-masalah. Misalkan, $Si(\cdot)$ dapat melakukan panggilan ke sistem IR, sehingga memungkinkan kita untuk mengakses rentang data yang lebih luas untuk pemecahan masalah. Contoh lain adalah menggunakan $Si(\cdot)$ sebagai kalkulator untuk menghitung hasil secara akurat dalam pemecahan masalah matematika. Jika sub-masalah p_i rumit dan memerlukan beberapa langkah pemecahan masalah antara, maka dimungkinkan juga untuk menguraikan lebih lanjut p_i menjadi sub-masalah yang lebih kecil. Misalkan, $Si(\cdot)$ dapat didefinisikan sebagai program rekursif yang menghasilkan dan memecahkan sub-masalah. Ini menggabungkan rekursi ke dalam pemecahan masalah dan memungkinkan kita untuk menangani masalah dengan mengurakannya secara berulang. Hasilnya, kita dapat mendefinisikan struktur hierarkis untuk pemecahan masalah [Khot et al., 2023].

Jika kita menggeneralisasikan rumusan di atas sedikit lebih jauh, kita dapat menganggapnya sebagai masalah pembelajaran penguatan. Metode yang umum adalah memodelkan proses pemecahan masalah sebagai proses pengambilan keputusan. Dalam setiap langkah proses ini, tindakan diambil berdasarkan keadaan saat ini. Tindakan ini

dapat mencakup semua fungsi untuk pembuatan dan penyelesaian sub-masalah (yaitu, $Gi(\cdot)$ dan $Si(\cdot)$). Dengan demikian, urutan tindakan sesuai dengan jalur penyelesaian masalah. Karena pembahasan masalah pembelajaran penguatan berada di luar cakupan bab ini, kami melewatkannya deskripsi yang tepat dari tugas pembelajaran ini. Meskipun demikian, mengembangkan agen atau pengontrol untuk menentukan kapan dan bagaimana membuat dan menyelesaikan sub-masalah juga merupakan pilihan yang wajar.

Dalam NLP, dekomposisi masalah terkait dengan serangkaian penelitian panjang tentang menjawab pertanyaan multi-hop [Mavi et al., 2024]. Tugas ini mengharuskan sistem untuk mengumpulkan dan menggabungkan informasi dari beberapa bagian teks untuk memberikan jawaban yang akurat untuk pertanyaan yang kompleks. Misalnya, untuk menjawab pertanyaan "Apa ibu kota negara tempat Albert Einstein lahir?", kita perlu mengetahui "Di mana Albert Einstein lahir?" dan "Apa ibu kota Jerman?". Pekerjaan sebelumnya di bidang ini dan yang terkait telah menyelidiki masalah dekomposisi masalah, meskipun metodenya mungkin tidak didasarkan pada LLM. Misalnya, metode yang populer adalah mengembangkan model saraf tambahan untuk menghasilkan pertanyaan yang lebih sederhana yang membahas berbagai aspek dari pertanyaan asli [Andreas et al., 2016; Talmor dan Berant, 2018; Min et al., 2019]. Generator pertanyaan ini dapat membuat pertanyaan secara batch atau berurutan.

Secara umum, dekomposisi masalah juga terkait dengan masalah komposisionalitas dalam NLP [Drozdov et al., 2022; Press et al., 2023]. Misalnya, dalam penguraian semantik, kami memetakan kalimat bahasa alami ke dalam representasi makna terstruktur dengan memecahnya menjadi bagian-bagian penyusun dan memahami kalimat berdasarkan makna bagian-bagian tersebut dan aturan yang digunakan untuk menggabungkannya. Dalam studi awal bidang ini, kalimat yang sangat komposisional dianggap lebih mudah untuk menguji sistem, karena relatif mudah untuk menguraikan kalimat tersebut dan menyusun makna bagian-bagiannya. Namun, tugas menjadi jauh lebih sulit ketika generalisasi yang lebih banyak diperlukan untuk memodelkan komposisionalitas dalam data baru. Dalam kasus ini, kami ingin sistem memiliki kemampuan generalisasi komposisional yang lebih baik. Dalam penelitian yang lebih baru tentang LLM, masalah ini telah sering dibahas dalam tugas penalaran komposisional, seperti SCAN⁷ karena dianggap sebagai aspek penting dalam menguji pemahaman bahasa dan kemampuan penalaran LLM. Ini juga menghadirkan tugas-tugas baru untuk mengembangkan dan memeriksa metode dekomposisi masalah.

Dalam LLM, satu aplikasi menarik dari dekomposisi masalah adalah penggunaan alat. Dalam beberapa kasus, perlu untuk mengintegrasikan alat eksternal ke dalam LLM untuk mengakses data akurat yang tidak tersedia selama pelatihan atau penyempurnaan. Misalnya, LLM dapat terintegrasi dengan API untuk mengambil data waktu nyata seperti pembaruan cuaca, harga pasar saham, atau umpan berita, yang memungkinkan mereka untuk memberikan respons terkini terhadap permintaan pengguna. Saat menggunakan alat, prediksi LLM mungkin menyertakan penanda yang menunjukkan di mana dan bagaimana memanggil API eksternal. Ini memerlukan penguraian masalah menjadi sub-masalah, dengan beberapa ditangani oleh LLM dan yang lainnya oleh alat eksternal. Pembahasan yang lebih rinci tentang topik ini akan disajikan di Bagian 3.2.5.

3.2.3 Penyempurnaan diri

Dalam banyak kasus, prediksi LLM bisa jadi tidak akurat atau salah. Mengingat bahwa LLM saat ini dapat melakukan tugas-tugas seperti penyempurnaan dan koreksi, masuk akal untuk mengeksplorasi metode bagi model-model ini untuk menyempurnakan sendiri keluarannya. Penyempurnaan sendiri merupakan fenomena umum dalam psikologi manusia

⁷Tugas SCAN (versi Sederhana dari tugas Navigasi CommAI) dirancang untuk mengevaluasi kemampuan LLM dalam melakukan generalisasi komposisi [Lake dan Baroni, 2018]. Tugas ini melibatkan penerjemahan perintah bahasa alami ke dalam serangkaian tindakan. Misalnya, perintah "lompat ke kiri dan berjalan tiga kali" dapat diterjemahkan ke dalam serangkaian tindakan "LTURN LTURN JUMP WALK WALK WALK".

aktivitas dan perilaku sehari-hari. Misalnya, saat mendesain sebuah produk, seorang desainer mungkin pertama-tama membuat prototipe dasar, lalu menyempurnakan desain setelah evaluasi dan pengujian untuk meningkatkan pengalaman dan fungsionalitas pengguna. Penyempurnaan dapat diulang beberapa kali hingga desainnya memuaskan. Ide prediksi-lalu-perbaiki juga dapat ditemukan dalam NLP. Salah satu contoh awal adalah tagger milik Brill [Brill, 1992], di mana hasil penandaan POS awal dari sebuah kalimat dapat disempurnakan secara berulang menggunakan sistem berbasis aturan. Di era pembelajaran mendalam, banyak pekerjaan pada masalah urutan-ke-urutan, seperti koreksi tata bahasa dan penulisan ulang teks, juga dapat

Kita dapat meminta LLM untuk melakukan penyempurnaan sendiri. Pertimbangkan contoh sederhana di mana kita pertama-tama menggunakan LLM untuk menerjemahkan teks dari bahasa Mandarin ke bahasa Inggris, lalu menggunakan model yang sama untuk menyempurnakan terjemahan. Berikut ini cara kita dapat meminta model untuk menerjemahkan:

Terjemahkan teks berikut dari bahasa Mandarin ke bahasa Inggris:

Cina:

perusahaan asuransi kesehatanyyyyyyyyyyyyyyyy,yyy5800yyy,yyyyyyyyyy
yyyyyyyyyy,yyyyyyyyyy

Bahasa

Indonesia: Serangkaian penemuan telah memperkuat peran penting budaya Hongshan dalam studi tentang asal-usul peradaban Tiongkok. Temuan penelitian terbaru menunjukkan bahwa, mulai sekitar 5.800 tahun yang lalu, banyak wilayah di seluruh Tiongkok mulai mengalami diferensiasi sosial yang nyata, memasuki fase percepatan dalam asal-usul peradaban.

Kemudian, kita dapat meminta untuk menyempurnakan terjemahannya:

Anda diberikan teks bahasa Mandarin beserta terjemahannya dalam bahasa Inggris.

Harap perbaiki terjemahan untuk meningkatkan akurasi dan kelancarannya jika perlu.

Cina:

perusahaan asuransi kesehatanyyyyyyyyyyyyyyyy,yyy5800yyy,yyyyyyyyyyyyyyyyyy
y,yyyyyyyyyy

Bahasa

Indonesia: Serangkaian penemuan telah memperkuat peran penting budaya Hongshan dalam studi tentang asal-usul peradaban Tiongkok. Temuan penelitian terbaru menunjukkan bahwa, mulai sekitar 5.800 tahun yang lalu, banyak wilayah di seluruh Tiongkok mulai mengalami diferensiasi sosial yang nyata, memasuki fase percepatan asal-usul peradaban.

Perbaikan:

Serangkaian penemuan arkeologi telah memperkuat peran penting budaya

Hongshan dalam studi tentang asal-usul peradaban Cina.

Temuan penelitian terbaru menunjukkan bahwa, mulai sekitar 5.800 tahun yang

lalu, berbagai wilayah di seluruh Tiongkok mulai mengalami diferensiasi sosial

yang nyata, memasuki fase percepatan dalam asal usul peradaban.

Di sini modifikasi pada terjemahan masukan disorot dengan warna hijau. Proses ini dapat bersifat iteratif: pada setiap langkah, kami menggunakan terjemahan dari langkah sebelumnya sebagai masukan, dan meminta LLM untuk menghasilkan terjemahan yang berpotensi lebih baik.

Dalam contoh di atas, penyempurnaan keluaran model awal dicapai dengan meminta LLM dengan instruksi sederhana seperti "Silakan perbaiki!". Akan tetapi, penyempurnaan hanya didasarkan pada kemampuan LLM dalam mengikuti instruksi, dan tidak ada panduan atau pengawasan tentang bagaimana dan di mana meningkatkan keluaran model. Pendekatan yang lebih efektif adalah dengan mempertimbangkan umpan balik pada aspek-aspek tertentu yang perlu disempurnakan. Misalnya, kita dapat meminta LLM dengan "Silakan perbaiki semua kesalahan tata bahasa dalam terjemahan", sehingga model dapat lebih fokus pada koreksi kesalahan tata bahasa selama penyempurnaan.

Kerangka umum penyempurnaan diri dengan LLM melibatkan tiga langkah [Madaan et al., 2024].

- **Prediksi.** Kami menggunakan LLM untuk menghasilkan keluaran model awal.
- **Pengumpulan Umpan Balik.** Kami memperoleh umpan balik atas keluaran model.
- **Penyempurnaan.** Kami menggunakan LLM untuk menyempurnakan keluaran model berdasarkan umpan balik.

Dua langkah terakhir dapat diulang beberapa kali, yang mengarah pada proses penyempurnaan diri yang berulang. Dalam kerangka kerja ini, menerima umpan balik yang akurat dan terperinci sangat penting karena hal ini memandu langkah penyempurnaan berikutnya. Secara umum, umpan balik dapat diperoleh dengan berbagai cara [Fernandes et al., 2023; Pan et al., 2024]. Kita dapat memeriksa prediksi model secara manual untuk mengidentifikasi masalah. Kita juga dapat menggunakan model umpan balik untuk menghasilkan umpan balik secara otomatis. Misalnya, model penghargaan yang dilatih pada beberapa data berlabel dapat membantu mengevaluasi keluaran model dan mengembalikan skor yang menunjukkan kualitasnya.

3.2 Metode Prompting Lanjutan

127

Karena fokus bab ini adalah pada dorongan LLM, kami mempertimbangkan penggunaan LLM untuk menyediakan umpan balik kembali. Sebagai ilustrasi, mari kita pertimbangkan contoh menghasilkan respons yang baik terhadap pertanyaan pengguna:

- Pertama, kami meminta LLM untuk menjawab pertanyaan masukan, seperti biasa.

Buatlah tanggapan terhadap pertanyaan: "Apa saja masalah lingkungan utama saat ini?"

Respons:

Masalah lingkungan utama saat ini meliputi perubahan iklim, polusi udara dan air, penggundulan hutan, hilangnya keanekaragaman hayati, dan degradasi laut.

- Kemudian, dengan menggunakan perintah berikut, kami meminta LLM yang sama lagi untuk mendapatkan umpan balik tentang respon yang dihasilkan.

Anda akan diberikan pertanyaan dan jawabannya. Evaluasi keakuratan dan kedalaman jawaban ini, dan identifikasi ketidakakuratan fakta atau area yang memerlukan penjelasan tambahan untuk meningkatkan pemahaman.

Pertanyaan:

Apa saja masalah lingkungan utama saat ini?

Respons:

Masalah lingkungan utama saat ini meliputi perubahan iklim, polusi udara dan air, penggundulan hutan, hilangnya keanekaragaman hayati, dan degradasi laut.

Masukan:

Respons yang diberikan akurat dan menyoroti berbagai masalah lingkungan utama. Akan tetapi, akan lebih baik jika ada uraian yang lebih spesifik dan terperinci tentang berbagai masalah tersebut. Berikut ini adalah beberapa saran untuk perbaikan:

1. Sertakan contoh spesifik masalah lingkungan.
2. Tambahkan informasi tentang polutan utama.
3. Tentukan wilayah yang paling terkena dampak.

- Mengingat umpan balik yang dihasilkan LLM, kami mendorongnya untuk menyempurnakan respons.

Anda akan diberikan pertanyaan, jawabannya, dan beberapa umpan balik mengenai jawaban ini.

Pertanyaan:

Apa saja masalah lingkungan utama saat ini?

Respons:

Masalah lingkungan utama saat ini meliputi perubahan iklim, polusi udara dan air, penggundulan hutan, hilangnya keanekaragaman hayati, dan degradasi laut.

Masukan:

Respons yang diberikan akurat dan menyoroti berbagai masalah lingkungan utama.

Akan tetapi, akan lebih baik jika ada uraian yang lebih spesifik dan terperinci tentang masalah-masalah ini. Berikut ini adalah beberapa saran untuk perbaikan:

1. Sertakan contoh spesifik masalah lingkungan.
2. Tambahkan informasi tentang polutan utama.
3. Tentukan wilayah yang paling terkena dampak.

Sekarang perbaiki respons berdasarkan umpan balik yang diberikan.

Respons yang

Disempurnakan: Isu-isu lingkungan utama meliputi perubahan iklim, dibuktikan

dengan meningkatnya suhu global dan pola cuaca yang buruk; polusi udara dan air

yang signifikan di daerah perkotaan dan industri; penggundulan hutan yang

merajalela di wilayah seperti Amazon, yang menyebabkan hilangnya keanekaragaman

hayati; dan degradasi laut, yang disorot oleh pemutihan terumbu karang dan penangkapan ikan berlebihan yang mel

Idealnya, jika LLM yang kuat diadopsi, kami ingin LLM tersebut menjalankan ketiga langkah tersebut tanpa pelatihan tambahan. Di sisi lain, jika kami memiliki cukup data berlabel untuk tugas yang diinginkan, kami dapat meningkatkan kinerja LLM menggunakan pembelajaran terbimbing. Misalnya, kami dapat menyempurnakan LLM agar lebih sesuai dengan tugas penyempurnaan, atau sebagai alternatif, menggunakan model khusus tugas, yang mungkin tidak selalu didasarkan pada LLM [Welleck et al., 2023; Schick et al., 2023]. Dalam pengertian yang lebih luas, meningkatkan LLM untuk tugas penyempurnaan diri dapat dilihat sebagai masalah penyelarasan. Misalnya, telah ditemukan bahwa beberapa kemampuan koreksi diri dapat diaktifkan melalui RLHF [Ganguli et al., 2023]. Namun, pembahasan masalah ini berada di luar cakupan bab ini. Pembahasan lebih lanjut dapat ditemukan di Bab 4.

Dalam LLM, penyempurnaan diri terkait dengan beberapa konsep yang menyingkapkan aspek psikologis model-model ini, seperti kemampuan untuk merefleksikan diri. Pandangan yang ada adalah bahwa jika LLM mampu melakukan refleksi diri, prediksi mereka dapat menjadi lebih akurat dan bahkan memiliki kemampuan untuk mengoreksi diri sendiri. Refleksi diri ini dapat diaktifkan dengan berbagai cara, misalnya, dengan mendorong LLM ini untuk terlibat dalam pemikiran yang lebih mendalam dan cermat, atau dengan memberikan contoh-contoh yang dapat dipelajari dan direfleksikan oleh model. Sebagai ilustrasi, kami mempertimbangkan di sini metode deliberate-then-generate (DTG) yang disajikan dalam karya Li et al. [2023a], di mana LLM diminta untuk berunding. Dalam DTG, kami diberikan keluaran model awal yang mungkin mengandung kesalahan. LLM kemudian diminta untuk mengidentifikasi jenis kesalahan dari keluaran model ini dan memberikan keluaran yang lebih baik. Di bawah ini adalah templat permintaan DTG untuk tugas penerjemahan dari bahasa Mandarin ke bahasa Inggris.

Mengingat kalimat bahasa Mandarin: {üsumberý}

Terjemahan bahasa Inggrisnya adalah: {**targety**}

Harap deteksi terlebih dahulu jenis kesalahannya, lalu perbaiki terjemahannya.

Tipe Kesalahan:

Kami bermaksud untuk terlebih dahulu memprediksi jenis kesalahan (merah), lalu menghasilkan terjemahan yang lebih baik (biru). Proses pertimbangan ini dipandu oleh instruksi “Harap terlebih dahulu mendeteksi jenis kesalahan, lalu menyempurnakan terjemahan”. Instruksi ini mendorong LLM untuk terlebih dahulu terlibat dalam analisis yang cermat, lalu memberikan hasil yang lebih baik. Karena prediksi dan penyempurnaan jenis kesalahan dilakukan dalam satu kali LLM, metode ini menggabungkan kedua langkah umpan balik dan penyempurnaan menjadi satu proses.

Dalam prompt di atas, kami berasumsi bahwa LLM yang kami gunakan mampu meninjau terjemahan masukan dan mengidentifikasi jenis kesalahannya dengan benar. Namun, hal ini menimbulkan kesulitan baru karena model tersebut mungkin tidak pandai menemukan kesalahan dalam terjemahan. Hal ini pada gilirannya akan menghasilkan upaya rekayasa penyempurnaan atau prompt-ing ekstra. Jadi, metode yang lebih sederhana adalah mengurangi beban identifikasi kesalahan dan menggunakan LLM hanya untuk pertimbangan. Untuk melakukan ini, kita dapat mengganti terjemahan masukan dengan terjemahan acak dan menetapkan jenis kesalahan default. Contoh prompt tersebut ditunjukkan di bawah ini:

Mengingat kalimat dalam bahasa Mandarin:

Terjemahan bahasa Inggrisnya

adalah: Berbagai teknik inovatif telah mendefinisikan ulang pentingnya seni modern dalam studi budaya kontemporer.

Harap deteksi terlebih dahulu jenis kesalahannya, lalu perbaiki terjemahannya.

Tipe Kesalahan: Terjemahan Salah

Dalam contoh ini, terjemahan masukan tidak dihasilkan oleh LLM tetapi diambil sampelnya secara acak dari kumpulan data. Jadi, ini hanyalah terjemahan yang salah untuk kalimat sumber, dan kita dapat mengatur jenis kesalahan yang sesuai. LLM kemudian menghasilkan terjemahan baru dengan mengambil kalimat sumber dan terjemahan yang salah sebagai masukan. Desain prompt ini juga dapat dianggap sebagai pengaktifan kemampuan belajar LLM melalui "bukti negatif" [Marcus, 1993], sehingga memungkinkan mereka untuk merefleksikan dan menghasilkan hasil yang lebih baik melalui analisis kontrastif. Meskipun demikian, metode ini tidak bergantung pada umpan balik apa pun dan dapat meningkatkan kinerja prediksi LLM tunggal melalui prompt sederhana.

Perhatikan bahwa meskipun DTG bersifat non-iteratif, pembelajaran dan penyempurnaan iteratif umumnya digunakan dalam NLP. Keuntungan dari pendekatan iteratif ini adalah meniru pembelajaran dan pemecahan masalah manusia, di mana umpan balik dan penyesuaian berkelanjutan menghasilkan hasil yang semakin baik. Metode iteratif dapat diterapkan pada berbagai masalah yang memicu LLM. Misalnya, dalam dekomposisi masalah, seseorang dapat memasukkan sub-masalah baru dan solusinya ke dalam konteks di setiap langkah, dan dengan demikian LLM dapat secara progresif mendekati solusi dari masalah asli.

Di sisi lain, metode iteratif menimbulkan beberapa masalah yang tidak terdapat pada metode non-iteratif, misalnya, kesalahan pada langkah awal dapat berdampak negatif pada pemecahan masalah selanjutnya, dan penentuan kapan harus menghentikan iterasi sering kali memerlukan upaya rekayasa tambahan.

3.2.4 Menyusun

Penggabungan model untuk pembuatan teks telah dibahas secara luas dalam literatur NLP. Ideanya adalah menggabungkan prediksi dari dua atau lebih model untuk menghasilkan prediksi yang lebih baik. Teknik ini dapat langsung diterapkan pada LLM. Misalnya, kita dapat mengumpulkan sekumpulan LLM dan menjalankan masing-masingnya pada input yang sama. Output akhir adalah prediksi gabungan dari model-model ini.

Untuk prompting LLM, kinerja juga dapat ditingkatkan dengan menggabungkan prediksi berdasarkan prompt yang berbeda. Misalkan kita memiliki LLM dan kumpulan prompt yang membahas tugas yang sama. Kita dapat menjalankan LLM ini dengan masing-masing prompt lalu menggabungkan prediksi.

Misalnya, berikut ini tiga templat perintah berbeda untuk penyederhanaan teks.

Jadikan teks ini lebih
sederhana. {*ÿtextÿ*}

Padatkan dan sederhanakan teks ini.
{*ÿtextÿ*}

Tulis ulang agar mudah
dibaca. {*ÿtextÿ*}

Masing-masing petunjuk ini akan mengarah pada prediksi yang berbeda, dan kita dapat mempertimbangkan ketiga prediksi untuk menghasilkan prediksi akhir.

Secara formal, misalkan $\{x_1, \dots, x_K\}$ adalah K perintah untuk melakukan tugas yang sama. Dengan LLM $\Pr(\cdot| \cdot)$, kita dapat menemukan prediksi terbaik untuk setiap x_i menggunakan $y^i = \arg \max_{y_i} \Pr(y_i | x_i)$. Prediksi ini dapat digabungkan untuk membentuk prediksi “baru”:

$$\hat{y} = \text{Gabungkan}(y^1, \dots, y^K) \quad (3.6)$$

Di sini $\text{Combine}(\cdot)$ adalah model kombinasi, yang dapat dirancang dengan beberapa cara berbeda. Misalnya, kita dapat memilih prediksi terbaik dengan pemungutan suara atau dengan mengidentifikasi prediksi yang paling tumpang tindih dengan prediksi lainnya. Metode lain untuk kombinasi model adalah dengan melakukan perataan model selama prediksi token. Misalkan y^j adalah token yang diprediksi pada langkah ke- j untuk kombinasi model.

probabilitas untuk memprediksi \hat{y}^j diberikan oleh

$$\hat{y}^j = \arg \max_{y^j} \log \Pr(y^j | x_k, y^1, \dots, y^{j-1}) \quad (3.7)$$

Dalam ensembling untuk LLM prompting, umumnya menguntungkan untuk menggunakan prompt yang beragam sehingga kombinasi tersebut dapat menangkap rentang respons potensial yang lebih luas. Praktik ini umum dalam pembelajaran ensemble, karena keberagaman membantu meratakan bias dan kesalahan yang mungkin khusus untuk setiap model atau konfigurasi tunggal. Dari sudut pandang Bayesian, kita dapat memperlakukan prompt x sebagai variabel laten, mengingat masalah yang menarik, p. Hal ini memungkinkan distribusi prediktif y yang diberikan p untuk ditulis sebagai distribusi $\Pr(y|x)$ yang dipunggirkan atas semua prompt yang mungkin

$$\text{Persamaan } \Pr(y|p) = \Pr(y|x) \Pr(x|p)dx \quad (3.8)$$

Integral menghitung total probabilitas y dengan mempertimbangkan semua nilai x yang mungkin, yang dibobot dengan kemungkinannya jika diberikan p . Di sini $\Pr(y|x)$ diberikan oleh LLM, dan $\Pr(x|p)$ adalah distribusi prior dari prompt untuk masalah tersebut. Ini adalah model yang baik karena integral secara efektif memperhitungkan ketidakpastian dalam pilihan x , memastikan bahwa distribusi prediktif akhir $\Pr(y|p)$ kuat dan mencakup semua variasi dan bias potensial dalam prompt. Namun, menghitung integral ini secara langsung dapat menjadi tidak layak secara komputasi karena ruang x yang berpotensi tak terbatas.

Salah satu pendekatan untuk mengatasi persoalan ini adalah dengan menggunakan metode seperti pengambilan sampel Monte Carlo, yang memperkirakan integral dengan menggunakan jumlah petunjuk yang terbatas dan dapat dikelola.

Meskipun perlakuan Bayesian didefinisikan dengan baik secara matematis, merupakan praktik umum dalam NLP untuk mengasumsikan prior yang tidak informatif atau seragam dan sebaliknya berfokus pada penyusunan serangkaian perintah yang beragam. Akibatnya, output dapat dihitung menggunakan model kombinasi langsung, seperti yang dijelaskan dalam Persamaan (3.6). Masalah dalam menciptakan perintah berkualitas tinggi dan beragam telah dipelajari dalam CoT dan area pembelajaran dalam konteks lainnya. Sebagian besar penelitian berfokus pada penggabungan berbagai contoh demonstrasi di berbagai perintah. Di sini, kami mencantumkan beberapa metode ini.

- Bilamana ada masalah, kami secara manual membuat sejumlah demonstrasi dan menggunakan demonstrasi yang berbeda-beda untuk perintah yang berbeda-beda.
- Jika diberi suatu masalah, kami menggunakan LLM untuk secara otomatis membuat demonstrasi dan petunjuk.
- Mengingat adanya perintah, kami membuat perintah yang berbeda dengan mengubah urutan demonstrasi di mengingatkan.
- Mengingat suatu perintah, kami menggunakan LLM untuk menghasilkan sejumlah perintah serupa.
- Mengingat suatu perintah, kita mengubahnya ke bentuk lain, misalnya menerjemahkannya ke bahasa lain.

Tentu saja, dalam praktiknya, kita dapat menggabungkan metode-metode ini untuk mencapai keragaman yang lebih besar. Asumsi yang mendasarinya adalah bahwa perintah yang beragam dapat menghasilkan keluaran model yang beragam. Hal ini khususnya terjadi ketika masalah yang kita hadapi relatif baru dan sulit. Untuk LLM yang lebih kuat dan lebih tangguh, varians dalam keluaran untuk perintah yang serupa mungkin tidak besar. Dalam kasus ini, manfaat melibatkan beberapa perintah bisa jadi sederhana.

Selain menyediakan beragam petunjuk untuk LLM, pendekatan lain adalah memanfaatkan varians inheren dalam keluaran LLM. Salah satu cara sederhana untuk menghasilkan beberapa keluaran adalah dengan mengambil sampel keluaran dari ruang hipotesis. Ini mudah untuk LLM karena algoritme pencarinya secara alami cocok untuk menghasilkan beberapa hasil. Misalnya, jika pencarian berkas digunakan, kita dapat dengan mudah mengumpulkan semua hipotesis lengkap dalam berkas untuk menghasilkan beberapa kandidat keluaran. Selain itu, kita dapat memodifikasi algoritme pencarian untuk menerapkan metode pengambilan sampel yang lebih canggih dan mengeksplorasi berbagai hipotesis berkualitas tinggi yang l

Pertanyaan berikutnya adalah bagaimana menggabungkan berbagai prediksi model. Hal ini membawa kita kembali ke masalah kombinasi model seperti yang dijelaskan dalam Persamaan (3.6). Di sini, kami mempertimbangkan metode konsistensi diri, yang tidak menghasilkan prediksi dengan probabilitas tertinggi, tetapi prediksi yang paling sesuai dengan prediksi lain [Wang et al., 2022a; 2023b]. Pertama, LLM diminta dengan CoT seperti biasa dan menghasilkan beberapa jalur penalaran dengan pengambilan sampel. Kemudian, frekuensi setiap jawaban dihitung di seluruh jalur penalaran ini. Keluaran akhir adalah jawaban yang memiliki probabilitas tertinggi.

Prediksi 1 (Benar):

T: Tiga orang teman bermain lempar koin. Setiap teman melempar koin satu kali.
Berapa peluang tepat satu dari mereka melempar gambar?

A: Mari kita berpikir selangkah demi selangkah.

Setiap lemparan koin merupakan kejadian independen dengan peluang 50% untuk mendapatkan sisi kepala (H) dan 50% sisi ekor (T). Pertimbangkan kemungkinan hasil untuk tiga lemparan: HHH, HHT, HTH, HTT, THH, THT, TTH, TTT. Dari semua itu, hanya HHH, HHT, HTH, dan THH yang relevan. Kami hanya tertarik pada skenario dengan tepat satu sisi kepala: HTT, THT, dan TTH. Ada 3 hasil seperti itu dari total 8, yang memberikan peluang 3/8.

Prediksi 2 (Benar):

T: Tiga orang teman bermain lempar koin. Setiap teman melempar koin satu kali.
Berapa peluang tepat satu dari mereka melempar gambar?

A: Mari kita berpikir selangkah demi selangkah.

Dengan menggunakan rumus probabilitas binomial, di mana p (sukses atau kepala) adalah 0,5, dan masing-masing dari 3 percobaan (flip) bersifat independen, kita mencari tepat satu kesuksesan (kepala). Rumus untuk tepat k kesuksesan dalam n percobaan diberikan oleh rumus koefisien binomial. Dengan $\binom{n}{k}$ memasukkan nilai-nilai tersebut, kita memperoleh

$$\binom{3}{1} (0,5)^1 (0,5)^{3-1} = 3 \cdot 0,5 \cdot 0,25 = 0,375.$$

Prediksi 3 (Salah):

T: Tiga orang teman bermain lempar koin. Setiap teman melempar koin satu kali. Berapa peluang tepat satu dari mereka melempar gambar?

A: Mari kita berpikir selangkah demi selangkah.

Karena setiap lemparan koin memiliki peluang 50% untuk mendapatkan sisi kepala dan peluang 50% untuk mendapatkan sisi ekor, dan kita mencari kasus di mana hanya satu dari tiga lemparan yang menghasilkan sisi kepala, kita mungkin awalnya berpikir ini hanya masalah memilih lemparan mana yang menghasilkan sisi kepala. Ada tiga lemparan, jadi seseorang mungkin secara keliru berasumsi bahwa peluang untuk memilih satu hasil tertentu seperti ini adalah 1 dari 3. Jadi, mereka mungkin menyimpulkan bahwa peluang untuk mendapatkan tepat satu sisi kepala adalah

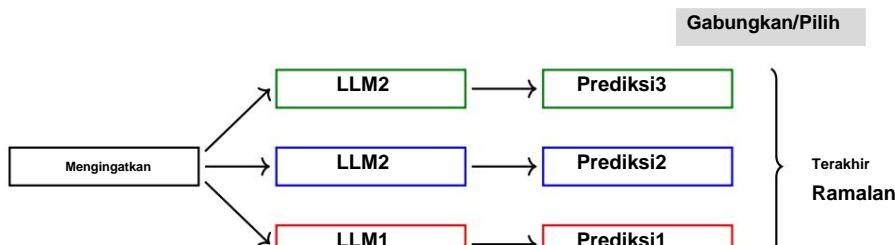
Prediksi 1 dan 2 mengidentifikasi tiga kasus dengan tepat di mana tepat satu kepala dilempar, keduanya memperoleh probabilitas sebesar 37,5%. Penalaran dalam Prediksi 3 gagal memperhitungkan jumlah total hasil yang mungkin terjadi dengan tiga lemparan koin, sehingga memberikan jawaban yang salah sebesar 33,3%. Oleh karena itu, kami memilih 37,5% sebagai jawaban akhir karena merupakan konsensus.

Konsistensi diri menyediakan kriteria untuk menentukan prediksi terbaik dalam kumpulan kandidat. Karena prompt dan model ditetapkan dalam metode ini, ini bukan metode ensembling prompt secara ketat. Sebaliknya, ini dapat dilihat sebagai contoh metode ensembling output, yang juga dikenal sebagai metode pemilihan hipotesis, yang telah lama dieksplorasi dalam NLP, khususnya untuk masalah pembuatan teks [Xiao et al., 2013]. Dalam metode ini, beberapa output dihasilkan dengan memvariasikan arsitektur model atau parameter. Setiap output kemudian diberi skor berdasarkan beberapa kriteria, dan output diperingkat ulang berdasarkan skor ini. Ada berbagai cara untuk menentukan fungsi penilaian, seperti mengukur kesepakatan antara output dan yang lain, dan menggunakan model yang lebih kuat untuk menilai ulang setiap output⁸. Gambar 3.2 menunjukkan perbandingan metode ensembling yang berbeda untuk LLM.

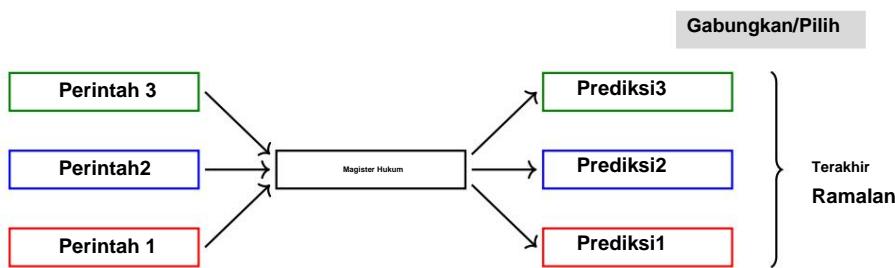
Sekarang, mari kita tinjau secara singkat metode-metode yang telah kita bahas sejauh ini di bagian ini, seperti dekomposisi masalah dan penyempurnaan diri. Jelas bahwa metode-metode ini meningkatkan pengambilan keputusan dengan memperkenalkan lebih banyak "pilihan" ke dalam proses penalaran. Sampai batas tertentu, semuanya melibatkan evaluasi dan pemberian umpan balik pada hasil LLM. Misalnya, dalam penyempurnaan diri, kita perlu menawarkan saran untuk meningkatkan prediksi LLM, dan dalam penyusunan keluaran, kita memilih keluaran optimal dari kumpulan kandidat. Dalam pengertian ini, metode-metode ini termasuk dalam kategori yang lebih luas dari pendekatan prediksi-lalu-verifikasi, di mana prediksi awalnya dibuat, kemudian diverifikasi dan disempurnakan. Masalah mendasar di sini melibatkan verifikasi dan evaluasi hasil penalaran atau langkah-langkah perantara. Masalah ini agak terkait dengan masalah pelatihan model penghargaan dalam RLHF, meskipun RLHF membahas aspek yang berbeda. Faktanya, pengembangan verifier telah dieksplorasi dan diimplementasikan dalam penalaran dengan LLM. Sebagian besar pekerjaan, daripada mengembangkan algoritma inferensi-waktu berbasis heuristik, berfokus pada pembelajaran verifier dengan cara yang diawasi. Metode yang mudah adalah melatih verifier sebagai pengklasifikasian biner, seperti mengklasifikasikan jawaban sebagai

⁸Interpretasi dari konsistensi diri adalah melihatnya sebagai proses pencarian risiko Bayes minimum. Proses ini mencari output terbaik dengan meminimalkan risiko Bayes. Lebih khusus lagi, fungsi risiko $R(y, yr)$ didefinisikan pada setiap pasangan output (dilambangkan dengan (y, yr)), yang mewakili biaya penggantian y dengan yr . Diberikan sekumpulan output \hat{y} , risiko output $y \neq \hat{y}$ diberikan oleh

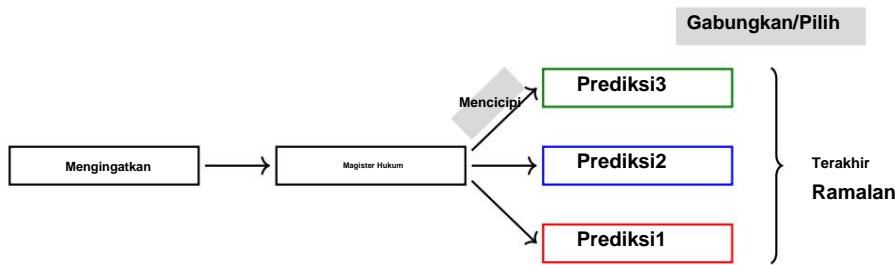
$$\begin{aligned} \text{Risiko}(y) &= E_{\hat{y}} \Pr(\hat{y}|x) R(y, \hat{y}) \\ &= \sum_{\text{tahun}\hat{y}} R(y, \hat{y}) \cdot \Pr(\hat{y}|x) \end{aligned} \tag{3.9}$$



(a) Model Ensembling



(b) Ensembling Cepat



(c) Output Ensembling

Gbr. 3.2: Metode ensembling untuk LLM. Dalam ensembling model standar (a), beberapa LLM yang bervariasi dalam arsitektur atau parameter digunakan. Setiap LLM menerima prompt yang sama dan menghasilkan prediksi. Prediksi ini digabungkan untuk menghasilkan prediksi akhir. Dalam ensembling prompt (b), kita memiliki satu LLM dan beberapa prompt. LLM menghasilkan prediksi untuk setiap prompt, dan prediksi ini digabungkan seperti biasa. Dalam ensembling output (c), LLM mengambil sampel beberapa prediksi di ruang prediksi yang diberikan prompt. Ini dapat dilihat sebagai metode untuk meningkatkan kinerja LLM itu sendiri. Perhatikan bahwa metode ensembling ini dapat digabungkan untuk meningkatkan keragaman prediksi. Misalnya, kita dapat menggunakan ensembling prompt dan ensembling output untuk mendapatkan p

benar atau salah, meskipun verifier ini biasanya digunakan sebagai model penilaian. Jika diberikan jalur penalaran untuk suatu masalah, verifier dapat digunakan untuk menilai seluruh jalur (disebut pendekatan berbasis hasil) [Cobbe et al., 2021], atau setiap langkah penalaran individual (disebut pendekatan berbasis proses) [Uesato et al., 2022; Lightman et al., 2024].

3.2.5 Penggunaan RAG dan Alat

RAG umumnya digunakan ketika LLM standar, yang hanya mengandalkan pengetahuan yang telah dilatih sebelumnya, kurang akurat dan mendalam dalam teks yang dihasilkan. Dengan mengambil dari basis data dan dokumen eksternal,

RAG dapat meningkatkan kualitas respons secara signifikan, memastikan respons tersebut relevan secara kontekstual dan benar secara faktual. Pendekatan semacam itu khususnya berguna dalam skenario yang memerlukan akurasi faktual tinggi dan informasi terkini, seperti menjawab pertanyaan yang rumit.

Konsep RAG telah disebutkan beberapa kali di bagian dan bab sebelumnya.

Untuk kelengkapan, kami uraikan langkah-langkah utama yang terlibat dalam RAG di sini.

- Kami menyiapkan koleksi teks yang diperlakukan sebagai sumber pengetahuan tambahan yang kami dapat mengakses.
- Kami mengambil teks yang relevan untuk kueri tertentu.
- Kami memasukkan teks yang diambil dan kueri ke dalam LLM, yang kemudian diminta untuk menghasilkan prediksi akhir.

Langkah 1 dan 2 dapat diimplementasikan dengan menggunakan sistem pencarian informasi eksternal. Misalnya, kita dapat menyimpan kumpulan teks dalam basis data vektor dan kemudian mengambil teks yang paling mirip melalui teknik pencarian berbasis vektor. Karena pencarian informasi bukan fokus bab ini, kita akan berasumsi bahwa sistem tersebut tersedia di pasaran dan menggunakanannya secara langsung.

Di sini kami menyajikan cara meminta LLM untuk menggunakan teks yang diambil. Sebagai ilustrasi, pertimbangkan contoh penggunaan LLM untuk menjawab pertanyaan berikut.

Di mana Olimpiade 2028 akan diadakan?

Kita cukup memasukkan pertanyaan ini ke mesin pencari online. Mesin pencari tersebut kemudian akan memberikan jawaban yang relevan. potongan teks yang ditemukan di internet, misalnya,

(Inggris)

Olimpiade Musim Panas 2028, secara resmi disebut Olimpiade XXXIV dan umum dikenal sebagai Los Angeles 2028 atau LA28, adalah acara multolahraga internasional mendatang yang dijadwalkan berlangsung dari tanggal 14-30 Juli 2028 di Amerika Serikat.

(Berita Olahraga)

Pada tahun 2028, Los Angeles akan menjadi kota ketiga, setelah London dan Paris, yang menjadi tuan rumah tiga Olimpiade setelah menjadi tuan rumah Olimpiade Musim Panas pada tahun 1932 dan 1984. Ini juga akan menjadi pertama kalinya Amerika Serikat menjadi tuan rumah Olimpiade sejak Olimpiade Musim Dingin 2002 di Salt Lake City.

...

Kita dapat menggunakan teks yang diambil ini sebagai konteks tambahan, dan meminta LLM untuk menghasilkan tanggapan berdasarkan teks-teks ini. Berikut adalah contoh perintah RAG.

Tugas Anda adalah menjawab pertanyaan berikut. Untuk membantu Anda, teks-teks yang relevan disediakan. Harap dasarkan jawaban Anda pada teks-teks ini.

Pertanyaan:

Di mana Olimpiade 2028 akan diadakan?

Teks Relevan 1:

Olimpiade Musim Panas 2028, secara resmi disebut Olimpiade XXXIV dan umumnya dikenal sebagai Los Angeles 2028 atau LA28 ...

Teks Relevan 2:

Pada tahun 2028, Los Angeles akan menjadi kota ketiga, setelah London dan Paris, yang menjadi tuan rumah tiga Olimpiade setelah ...

...

Olimpiade 2028 akan diadakan di Los Angeles.

Prompt ini mengasumsikan bahwa teks yang diberikan relevan dengan pertanyaan dan mengharapkan LLM untuk menghasilkan respons yang tepat menggunakan teks-teks ini. Namun, sistem pencarian informasi terkadang dapat memberikan teks yang tidak relevan atau tidak benar, yang dapat menyebabkan LLM menghasilkan jawaban yang salah. Salah satu cara langsung untuk mengatasi masalah ini adalah dengan meningkatkan keakuratan sistem pencarian informasi. Meskipun demikian, seperti kebanyakan sistem AI, kesalahan mungkin masih terjadi. Oleh karena itu, penting juga untuk meningkatkan ketahanan LLM, sehingga dapat membuat prediksi yang wajar bahkan ketika input tidak akurat. Di bawah ini adalah prompt baru yang memungkinkan LLM untuk lebih setia pada fakta, dan memungkinkannya untuk memilih untuk tidak menjawab pertanyaan ketika informasi yang diberikan tidak akurat.

Tugas Anda adalah menjawab pertanyaan berikut. Untuk membantu Anda, teks-teks yang relevan disediakan. Harap dasarkan jawaban Anda pada teks-teks ini.

Harap perhatikan bahwa jawaban Anda harus seakurat mungkin dan sesuai dengan fakta. Jika informasi yang diberikan tidak cukup untuk memberikan jawaban yang akurat, Anda cukup menuliskan "Tidak menjawab!".

Pertanyaan:

Di mana Olimpiade 2028 akan diadakan?

Teks Relevan 1:

Olimpiade Musim Panas 2024, yang secara resmi disebut Olimpiade XXXIII dan diberi merek Paris 2024, adalah acara multiolahraga internasional ...

...

Tidak ada Jawaban!

Dalam contoh ini, LLM menolak untuk menjawab karena informasi yang diberikan tidak memadai dan tidak relevan dengan pertanyaan.

Baik RAG maupun fine-tuning merupakan metode umum untuk mengadaptasi LLM menggunakan data tugas spesifik. RAG standar bebas pelatihan dan dapat langsung diterapkan pada LLM. Untuk lebih meningkatkan RAG,

juga memungkinkan untuk menyempurnakan LLM, meskipun ini akan memerlukan beberapa upaya pelatihan. Misalnya, kita dapat menyempurnakan LLM menggunakan data berlabel manusia untuk mengawasi mereka dalam belajar menolak menjawab. Perhatikan bahwa, meskipun contoh-contoh yang ditunjukkan di atas tampak sederhana, RAG bukanlah hal yang sepele. Dari perspektif rekayasa prompt, kasus penggunaan yang berbeda mungkin memerlukan prompt yang berbeda, meskipun tujuan kita yang agak "rakus" adalah untuk mengembangkan strategi prompt universal yang dapat beradaptasi dengan berbagai tugas. Dalam banyak kasus, kita perlu mengendalikan seberapa besar kita bergantung pada konteks yang diambil untuk membuat prediksi. Terkadang, LLM harus memperoleh respons hanya dari teks yang disediakan, sementara di waktu lain, mereka mungkin perlu menghasilkan respons menggunakan pengetahuan yang telah mereka peroleh sebelumnya jika teks yang disediakan tidak mencukupi. Ada banyak aspek RAG, seperti peningkatan sistem pengambilan, yang tidak dapat dibahas dalam bab ini. Pembaca yang tertarik dapat merujuk ke survei teknik RAG untuk informasi lebih lanjut [Li et al., 2022; Gao et al., 2023c].

Salah satu alasan mengapa kita membahas RAG di sini adalah karena secara umum RAG dapat dianggap sebagai contoh kerangka dekomposisi masalah umum (lihat Bagian 3.2.2). RAG membagi pemecahan masalah menjadi dua langkah. Pada langkah pertama, kita mengumpulkan informasi yang relevan dan mendukung untuk kueri tertentu dari berbagai sumber pengetahuan. Pada langkah kedua, kita menggunakan LLM untuk menghasilkan respons berdasarkan informasi yang dikumpulkan. Jika kita memperluas konsep dekomposisi masalah lebih jauh, kita akan menemukan bahwa banyak tugas yang memerlukan penggunaan sistem atau alat eksternal dapat diperlakukan sebagai masalah yang serupa. Salah satu contohnya adalah penggunaan alat dalam LLM. Dalam banyak aplikasi, LLM perlu menggunakan basis data eksternal, API, dan bahkan alat simulasi untuk menghasilkan respons yang akurat. Misalnya, LLM dapat mengakses data waktu nyata dari pasar keuangan untuk memberikan saran investasi terkini atau berintegrasi dengan basis data perawatan kesehatan untuk menawarkan wawasan medis yang dipersonalisasi. Integrasi ini memperluas kemampuan LLM dengan memungkinkan mereka berinteraksi dengan, dan dalam beberapa konteks, memengaruhi atau mengendalikan sistem eksternal. Oleh karena itu, LLM berfungsi lebih sebagai agen otonom daripada sekadar generator teks [Franklin dan Graesser, 1996].

Masalah penggunaan alat itu luas dan luas. Di sini kami mempersempit pembahasan kami ke tugas-tugas yang dapat difasilitasi dengan memanggil API eksternal untuk menyelesaikan beberapa sub-masalah [Parisi et al., 2022; Gao et al., 2023b]. Pertimbangkan lagi contoh meminta LLM untuk menjawab "Di mana Olimpiade 2028 akan diadakan?". Misalkan LLM dapat mengakses alat pencarian web. Kami kemudian dapat meminta LLM untuk menjawab pertanyaan dengan pencarian web, seperti ini

Tugas Anda adalah menjawab pertanyaan berikut. Anda dapat menggunakan alat bantu eksternal, seperti pencarian web, untuk membantu Anda.

Pertanyaan:

Di mana Olimpiade 2028 akan diadakan?

Informasi mengenai pertanyaan ini diberikan sebagai berikut: {tool: web-

search, query: "Olimpiade 2028"}

Jadi jawabannya adalah: Los Angeles

Di sini {tool: web-search, query: "Olimpiade 2028"} menunjukkan permintaan ke sistem pencarian web menggunakan query "Olimpiade 2028". Ketika LLM melihat string ini, ia menjalankan pencarian web dan menggunakan hasilnya untuk mengganti string tersebut. Kemudian, pada langkah prediksi berikutnya, LLM menggunakan hasil pencarian web ini sebagai konteks untuk menghasilkan jawaban yang benar.

Pertimbangkan contoh lain di mana kita meminta LLM untuk memecahkan masalah matematika.

Masalah:

Sebuah kolam renang perlu diisi dengan air. Kolam renang tersebut berukuran panjang 10 meter, lebar 4 meter, dan kedalaman 2 meter. Hitunglah volume kolam renang dalam meter kubik, lalu tentukan berapa liter air yang dibutuhkan untuk mengisinya (dengan asumsi 1 meter kubik sama dengan 1000 liter).

Larutan:

Untuk menyelesaikan soal ini, LLM perlu menghitung volume kolam terlebih dahulu dengan menggunakan rumus volume prisma persegi panjang: Panjang × Lebar × Kedalaman. Jadi, volumenya adalah $10 \text{ m} \times 4 \text{ m} \times 2 \text{ m}$ = {tool: calculator, expression: $10 * 4 * 2$ } m³. Selanjutnya, untuk mencari tahu berapa liter air yang dibutuhkan, LLM mengalikan volume dalam meter kubik dengan 1000 (karena 1 meter kubik sama dengan 1000 liter). Jadi, 80×1000 = {tool: calculator, expression: $80 * 1000$ } liter.

Di sini, string {tool: calculator, expression: $10 * 4 * 2$ } memicu pemanggilan interpreter matematika untuk menghitung hasil ekspresi. Perhatikan bahwa hasilnya (yaitu, 80) akan mengantikan {tool: calculator, expression: $10 * 4 * 2$ } dan dapat dirujuk dalam prediksi token berikut. Misalnya, pada langkah terakhir pemecahan masalah, 80 digunakan sebagai ganti {tool: calculator, expression: $10 * 4 * 2$ }.

Perbedaan utama antara contoh penggunaan alat di sini dan contoh RAG yang dibahas sebelumnya adalah bahwa dalam penggunaan alat, fungsi eksternal dapat dipanggil selama inferensi. Sebaliknya, dalam RAG, teks yang diambil disediakan sebelum proses prediksi dimulai. Namun, dari perspektif pemodelan bahasa, keduanya sebenarnya melakukan hal yang sama: sebelum menghasilkan hasil akhir, kami menggunakan alat eksternal, baik secara manual maupun otomatis, untuk memperoleh konteks yang memadai dan relevan. Interpretasi tingkat tinggi dari pendekatan ini adalah bahwa keduanya bergantung pada "agen" yang dapat menentukan di mana dan bagaimana memanggil fungsi eksternal untuk menghasilkan konteks yang diperlukan.

Masalah dengan penggunaan alat adalah bahwa LLM asli tidak dilatih untuk menghasilkan penanda yang diperlukan untuk penggunaan alat. Oleh karena itu, kita perlu menyempurnakan LLM untuk mengadaptasinya untuk tugas-tugas ini [Schick et al., 2024]. Karena bab ini berfokus pada prompting, kita tidak akan menyajikan detail dari proses penyempurnaan ini. Sederhananya, pertama-tama kita perlu membuat anotasi pada data. Untuk setiap contoh penyempurnaan, kita mengganti bagian dari output yang memerlukan penggunaan alat eksternal dengan perintah atau penanda yang telah ditentukan sebelumnya. Kemudian, kita menggunakan data berlabel ini untuk menyempurnakan parameter LLM seperti biasa. Hasilnya, LLM dapat memperoleh kemampuan untuk menghasilkan perintah untuk memanggil alat eksternal. Selama inferensi, kita dapat menjalankan perintah penggunaan alat ini dalam output model untuk mendapatkan bantuan dari alat eksternal.

3.3 Belajar Memberi Petunjuk

Sejauh ini dalam bab ini, kita telah mempertimbangkan beberapa strategi dasar prompting dan berbagai penyempurnaan terhadapnya. Namun, semua prompt yang telah kita bahas dirancang secara manual. Hal ini menimbulkan sejumlah masalah: Pertama, merancang prompt berkualitas tinggi pada dasarnya sulit dan memerlukan upaya manual yang substansial. Misalnya, eksperimen ekstensif dengan prompt yang berbeda sering kali diperlukan untuk mengidentifikasi prompt yang paling efektif. Karena LLM yang berbeda mungkin merespons lebih baik terhadap prompt yang berbeda.

jenis perintah, pengembangan perintah yang efektif secara universal dapat memerlukan sumber daya yang lebih intensif. Kedua, desain perintah manual sangat bergantung pada keahlian manusia, yang dapat membatasi keberagaman pendekatan dan mengabaikan perintah yang berpotensi efektif yang tidak langsung terlihat oleh manusia. Ketiga, perintah yang dibuat oleh manusia bisa jadi rumit dan berulang-ulang, yang menyebabkan input menjadi lebih panjang. untuk LLM dan biaya komputasi yang lebih tinggi.

Pada bagian ini, kami membahas teknik untuk prompt otomatis. Metode ini bertujuan untuk membuat, mengoptimalkan, dan merepresentasikan prompt secara otomatis sehingga tugas-tugas selanjutnya dapat ditangani. lebih efektif dan efisien. Secara khusus, kami mempertimbangkan tiga isu di sini.

- Bagaimana kita dapat mengotomatiskan proses mendesain dan mengoptimalkan perintah untuk LLM?
- Apakah ada bentuk lain yang mewakili perintah selain string, dan bagaimana kita bisa mempelajarinya? representasi?
- Bagaimana kita bisa membuat perintah lebih ringkas dan padat, sehingga mengurangi kompleksitasnya dan panjang?

Perlu dicatat bahwa ada banyak pengaturan di mana kita dapat menyelidiki masalah ini. Misalnya, kita mungkin menentukan bahwa prompt dikembangkan secara khusus untuk LLM tertentu, atau bahwa pengembangannya independen dari LLM yang digunakan. Pengaturan ini dapat mengarah pada metode dan aplikasi yang berbeda. skenario, tetapi metode ini mungkin tumpang tindih dalam beberapa hal. Dalam diskusi berikut, kita akan membahas beberapa skenario berbeda dan mendiskusikan hubungan antara berbagai metode.

3.3.1 Optimasi Prompt

Mengingat desain yang cepat itu sulit dan membutuhkan banyak tenaga kerja, sebaiknya menggunakan pembelajaran mesin model untuk menemukan prompt optimal untuk tugas tertentu (sebut saja desain prompt otomatis atau Pendekatan ini secara luas dapat dianggap sebagai contoh pembelajaran mesin otomatis (AutoML), yang bertujuan untuk mengurangi atau menghilangkan kebutuhan akan manual yang digerakkan oleh ahli. desain model pembelajaran mesin. Meskipun fokus kita di sini adalah pada desain prompt, prompt sendiri merupakan struktur diskrit. Oleh karena itu, merancang perintah sangat mirip dengan merancang model pembelajaran mesin, seperti arsitektur model diskrit. Mungkin salah satu bidang yang paling terkait adalah pencarian arsitektur saraf (NAS), di mana jaringan saraf paling optimal diidentifikasi oleh menjelajahi ruang jaringan saraf yang mungkin [Zoph dan Le, 2016; Elsken et al., 2019]. Jika kita mempertimbangkan optimasi prompt sebagai proses pencarian, maka kita dapat menggambarkan optimasi prompt umum kerangka kerja yang melibatkan komponen-komponen berikut:

- **Prompt Search Space.** Ini mendefinisikan semua kemungkinan prompt yang dapat ditelusuri oleh algoritma. Misalnya, seseorang dapat mengedit beberapa prompt benih untuk menghasilkan sekumpulan prompt kandidat yang beragam.
- **Estimasi Kinerja.** Setelah suatu prompt dipilih, prompt tersebut perlu dievaluasi. Misalnya, cara mudahnya adalah memasukkannya ke LLM dan mengukur kinerjanya pada validasi mengatur.
- **Strategi Pencarian.** Proses pencarian umumnya sama dengan yang digunakan di banyak sistem AI. Pada setiap langkah, sistem mengeksplorasi serangkaian petunjuk yang menjanjikan di ruang pencarian dan

mengevaluasinya. Proses ini berlanjut saat lebih banyak prompt dieksplorasi. Hasil pencarian adalah prompt dengan performa terbaik yang diamati hingga pencarian berhenti.

Ini adalah kerangka kerja yang sangat umum, dan sistem pengoptimalan prompt yang berbeda dapat bervariasi dalam desain masing-masing komponen. Pendekatan yang banyak digunakan adalah menggunakan LLM sebagai dasar untuk mengembangkan komponen-komponen ini. Awalnya, beberapa prompt disediakan. Kemudian, proses berikut diulang hingga kriteria penghentian terpenuhi: 1) prompt dievaluasi pada set validasi; 2) kumpulan kandidat dipertahankan dengan hanya menyimpan prompt yang paling menjanjikan; dan 3) prompt baru dibuat dengan menggunakan LLM untuk menyimpulkan prompt serupa dari kumpulan kandidat ini. Salah satu manfaat dari pendekatan ini adalah memungkinkan kita untuk menggunakan LLM siap pakai untuk melakukan tugas-tugas yang disebutkan di atas tanpa perlu pengembangan sistem yang substansial. Untuk mencapai ini, kita dapat meminta atau menyempurnakan LLM untuk menyesuaikannya dengan tugas-tugas ini.

Di sini kita mempertimbangkan metode Zhou et al. [2023c] untuk mengilustrasikan pengoptimalan prompt berbasis LLM. Ini melibatkan

- Inisialisasi. Misalkan C mewakili kumpulan prompt kandidat yang ingin kita jelajahi. Langkah pertama adalah menambahkan prompt awal ke dalam C . Kita dapat melakukannya dengan beberapa cara. Metode yang sederhana adalah membuat prompt tersebut secara manual untuk tugas tertentu. Namun, dalam banyak kasus di mana manusia memiliki pengetahuan terbatas tentang cara menulis prompt yang efektif untuk tugas tersebut, mengembangkan prompt menjadi tantangan. Dalam kasus ini, sebaiknya gunakan LLM untuk membuat prompt. Misalnya, kita dapat langsung memberi instruksi kepada LLM untuk membuat perintah dan memberi mereka deskripsi tugas.

Anda diberi tugas untuk diselesaikan menggunakan LLM. Harap tulis petunjuk untuk memandu LLM.

{\deskripsi-tugas}

Metode ini mudah, tetapi masih memerlukan deskripsi tugas yang diberikan manusia. Metode alternatif adalah menggunakan LLM untuk membuat perintah berdasarkan contoh masukan dan keluaran tugas. Berikut ini adalah contoh perintah.

Anda diberikan beberapa pasangan input-output untuk suatu tugas. Harap tulis instruksi untuk melakukan tugas ini.

Masukan: {\input1} Keluaran: {\output1}

Masukan: {\input2} Keluaran: {\output2}

...

Dengan demikian, LLM dapat menyimpulkan instruksi yang sesuai untuk tugas dari masukan dan keluaran yang diberikan.

- Evaluasi. Setelah kita memperoleh kumpulan kandidat C , kita perlu mengevaluasi perintah-perintah di C . Salah satu metode adalah memasukkan setiap prompt ke dalam LLM dan menilai hasilnya di hilir

tugas. Misalnya, kita dapat mengevaluasi output LLM yang diberikan input menggunakan metrik, atau sebagai alternatif, menggunakan log-likelihood dari output sebagai ukuran kualitas perintah.

- **Pemangkasan.** Jika C berisi sejumlah besar prompt, maka pemangkasan prompt yang tidak menjanjikan adalah hal yang wajar. perintah di dalamnya, sehingga mengurangi beban komputasi pada langkah selanjutnya. Ini adalah masalah pemangkasan standar. Mengingat skor evaluasi untuk setiap prompt, metode sederhana adalah untuk menyimpan hanya persentase tertentu dari perintah tersebut dan membuang sisanya.
- **Ekspansi.** Ekspansi adalah operasi kunci dalam algoritma pencarian yang digunakan untuk menjelajahi berbagai negara bagian dalam ruang pencarian. Operasi ekspansi di sini dapat didefinisikan sebagai fungsi

$$C' = \text{Memperluas}(C, f) \quad (3.10)$$

dimana C' adalah kumpulan perintah baru yang dihasilkan dari C menggunakan model f . Jika kita mempertimbangkan f sebagai LLM, kita dapat melakukan operasi ekspansi dengan menginstruksikan f untuk menghasilkan yang baru dan petunjuk relevan berdasarkan C . Berikut adalah contohnya.

Berikut ini adalah perintah untuk LLM. Harap berikan beberapa perintah baru untuk melakukan tugas yang sama.

Masukan: {yprompt}

Kemudian kita mengganti C dengan C' . Tahapan evaluasi, pemangkasan dan perluasan dapat diulang kembali gambut, sehingga kita dapat secara bertahap menjelajahi rentang petunjuk yang lebih luas.

Dalam optimasi cepat, langkah ekspansi memainkan peran penting, karena ini mendefinisikan cara kita mengeksplorasi ruang pencarian, dan tujuan kami adalah menemukan hasil optimal dengan usaha minimal. Satu peningkatan langkah ini adalah memperlakukan masalah sebagai tugas parafrase. Metode sederhana adalah menerapkan sistem parafrase siap pakai, baik berdasarkan LLM atau model lain, untuk mengubah perintah masukan ke dalam bentuk yang setara secara semantik [Jiang et al., 2020]. Atau, kita dapat mendefinisikan edit spesifik operasi, seperti penyisipan dan modifikasi, untuk setiap token. Perintah yang diberikan dapat diedit ke dalam prompt baru dengan menerapkan operasi ini [Prasad et al., 2023]. Selain itu, evaluasi lebih lanjut dan pemangkasan dapat diterapkan untuk menyaring prompt berkualitas rendah. Selain pembuatan prompt pembingkaian sebagai masalah parafrase, kita dapat meningkatkan kualitas prompt selama ekspansi dengan mempelajari dari umpan balik [Pryzant et al., 2023]. Pendekatan ini agak terkait dengan penyempurnaan diri masalah yang dibahas di Bagian 3.2.3. LLM dapat digunakan untuk menghasilkan umpan balik pada permintaan masukan, yang kemudian direvisi berdasarkan umpan balik ini. Siklus umpan balik dan revisi ini dapat diulang beberapa kali hingga hasilnya sesuai atau keluaran yang diinginkan tercapai.

Pendekatan lain untuk optimasi cepat adalah dengan menerapkan teknik optimasi klasik. misalnya, permasalahan ini dapat dibingkai sebagai permasalahan komputasi evolusioner, dimana perintah diperlakukan sebagai kandidat yang berkembang dari generasi ke generasi seiring dengan kemajuan optimasi [Guo et al., 2024]. Karena banyak algoritma optimasi yang kuat telah dikembangkan dalam hal terkait bidang, mereka dapat langsung diterapkan pada masalah ini.

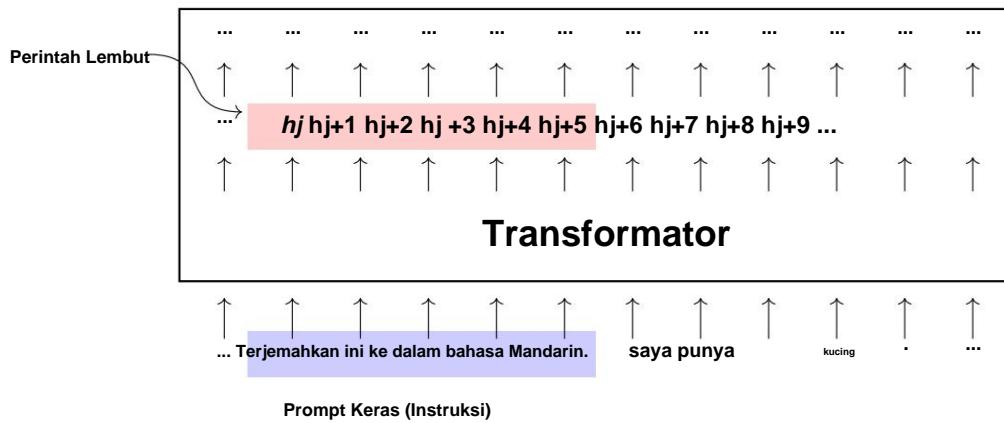
Dalam praktiknya, kita mungkin tergoda untuk menggunakan API LLM yang ada untuk mengimplementasikan langkah-langkah yang dijelaskan di atas. Namun, pendekatan semacam itu akan sangat bergantung pada kemampuan inferensi dan pembelajaran dalam konteks LLM. Jika LLM ini tidak kuat dan kurang beradaptasi dengan tugas, LLM tersebut dapat menimbulkan kesalahan dalam pencarian, misalnya, menghasilkan perintah yang salah selama perluasan. Dalam kasus seperti itu, lebih baik melatih model yang lebih sesuai dengan tugas. Salah satu pendekatan dalam arah penelitian ini mengacu pada pembelajaran penguatan, yang telah banyak digunakan dalam memecahkan masalah pengambilan keputusan dan pengoptimalan diskrit. Misalnya, [Deng dkk. \[2022\]](#) mengembangkan generator perintah dengan mengintegrasikan adaptor berbasis FFN ke dalam LLM. Generator perintah dilatih sebagai jaringan kebijakan umum, tetapi hanya parameter adaptor yang diperbarui sementara parameter model yang tersisa tetap tidak berubah. Selama pelatihan, imbalan diperoleh dengan menguji perintah yang dihasilkan menggunakan LLM lain, mirip dengan metode evaluasi seperti yang dibahas di atas. Setelah pelatihan selesai, generator perintah kemudian digunakan untuk menghasilkan perintah baru.

Perhatikan bahwa, dalam pembahasan kita di sini, perintah hanya dilihat sebagai urutan token, dan hasil dari pengoptimalan perintah adalah urutan tersebut. Namun, dalam arti sempit, perintah memiliki struktur yang kompleks dan mencakup berbagai bidang seperti masukan pengguna, instruksi, dan demonstrasi. Sementara pendekatan yang kita bahas sebagian besar bersifat umum, banyak pekerjaan dalam pengoptimalan perintah difokuskan pada mempelajari instruksi yang lebih baik untuk perintah. Secara khusus, tujuannya adalah untuk menghasilkan instruksi yang secara efektif memandu LLM berdasarkan tugas yang diberikan. Tentu saja, konsep pengoptimalan perintah juga dapat diperluas untuk mempelajari bagian lain dari perintah. Misalnya, ada minat penelitian yang substansial dalam mempelajari cara memilih atau menghasilkan demonstrasi di CoT [\[Liu et al., 2022; Rubin et al., 2022; Zhang et al., 2023b\]](#). Salah satu perbedaan antara mempelajari instruksi dan mempelajari demonstrasi adalah bahwa menghasilkan demonstrasi berkualitas tinggi menggunakan LLM relatif mudah dan fokus mempelajari demonstrasi biasanya pada cara mengambil sampel demonstrasi yang sesuai dari kumpulan kandidat. Sebaliknya, kesulitan dalam mempelajari instruksi sebagian disebabkan oleh LLM yang telah dilatih sebelumnya tidak cocok untuk memprediksi kualitas instruksi, dan pengujian instruksi ini pada tugas hilir memerlukan komputasi yang mahal. Hal ini membuat metode pengoptimalan mahal untuk diterapkan, dan mengeksplorasi berbagai macam instruksi menimbulkan tantangan yang signifikan.

3.3.2 Perintah Lembut

Meskipun mengembangkan perintah bahasa alami, baik secara manual maupun otomatis, merupakan pendekatan yang mudah dan banyak diterapkan, pendekatan ini memiliki beberapa masalah. Salah satu masalahnya adalah perintah bahasa alami bisa rumit dan panjang, sehingga menimbulkan beban komputasi yang signifikan saat diproses melalui LLM. Dalam banyak aplikasi, pengguna mungkin perlu melakukan tugas berulang kali, dan memasukkan perintah panjang yang sama ke dalam LLM berkali-kali jelas tidak efisien. Masalah lainnya adalah bahwa meskipun prompt biasanya direpresentasikan sebagai urutan token diskret (sebut saja prompt keras) dalam input LLM biasa, LLM mengodekannya sebagai vektor bernilai riil berdimensi rendah. Hal ini menimbulkan pertanyaan apakah ada cara yang lebih ringkas dan efisien untuk merepresentasikan prompt.

Dalam subbagian ini, kami memperkenalkan konsep soft prompt, yang dapat dilihat sebagai representasi prompt yang tersembunyi dan terdistribusi. Saat meminta LLM, kami berkepentingan dengan mengomunikasikan tugas atau pertanyaan untuk memperoleh respons yang diinginkan. Kami dapat mendefinisikan hard prompt sebagai urutan teks eksplisit dan terdefinisi sebelumnya yang dimasukkan pengguna secara langsung ke LLM untuk memandu respons. Sebaliknya, kami dapat menganggap soft prompt sebagai pola permintaan. Tidak seperti perintah keras, yang diekspresikan dalam bahasa alami dan harus dapat dipahami oleh



Gambar 3.3: Ilustrasi prompt keras dan lunak. Di sini, prompt keras adalah instruksi yang kita masukkan ke LLM untuk melakukan tugas. LLM mengkodekan instruksi ini seperti biasa, dan representasi perantara yang sesuai dengan instruksi tersebut dapat dilihat sebagai semacam perintah lunak.

manusia, perintah lunak dikodekan dalam format yang lebih mudah dipahami oleh model daripada manusia. Sebagai ilustrasi, perhatikan perintah sederhana

Terjemahkan kalimat berikut ke dalam bahasa Mandarin.

Anggap saja sudah selesai!

Di sini, instruksi “Terjemahkan kalimat ke dalam bahasa Mandarin” dapat dilihat sebagai perintah keras, yang dilambangkan dengan urutan token $c_1 \dots c_5$. Dengan memasukkan token-token ini ke dalam LLM, token-token ini akan diubah menjadi urutan vektor bernilai riil $h_1 \dots h_5$, yang masing-masing sesuai dengan token. Secara kasar kita dapat menganggapnya sebagai $h_1 \dots h_5$ sebagai soft prompt, seperti diilustrasikan dalam Gambar 3.3.

Meskipun contoh di atas menunjukkan bahwa soft prompt dapat dibuat dengan mengubah hard prompt menjadi soft prompt. petunjuk, belum tentu ada korespondensi langsung di antara mereka. Bahkan, kita bahkan tidak perlu menginterpretasikan perintah lunak menggunakan teks yang bermakna. Perintah tersebut hanyalah keadaan tersembunyi dalam LLM dan dapat dipelajari sebagai parameter standar model melalui optimasi berkelanjutan. Perlakuan seperti itu memungkinkan kita untuk mengeksplorasi metode prompting di luar teks. Sebagai manfaat lainnya, prompt lunak memberikan representasi yang padat, berdimensi rendah, dan dapat dipelajari untuk mengkodekan bagaimana kami memandu LLM untuk menghasilkan output tertentu. Pelatihan dan penerapan representasi ini memerlukan biaya komputasi yang jauh lebih rendah daripada yang diperlukan untuk memproses data yang panjang dan sulit. Pendekatan ini akan sangat bernilai praktis dalam aplikasi inferensi LLM di mana perintah yang sama digunakan berulang kali.

3.3.2.1 Mengadaptasi LLM dengan Lebih Sedikit Dorongan

Salah satu cara yang jelas untuk mengadaptasi LLM untuk tugas tertentu adalah dengan menyempurnakan model menggunakan data berlabel. Hal ini mengarah pada berbagai metode penyelarasan LLM, seperti fine-tuning yang diawasi, yang memperbarui parameter model dengan menyelaraskan respons terhadap perintah yang diberikan dengan pengawasan sinyal. LLM yang disetel dengan baik menanamkan informasi terkait tugas dalam parameter model, dan dengan demikian ini

model dapat merespons dengan benar saat menangani perintah yang serupa dengan perintah dalam penyetelan halus.

Jika kita mengembangkan ide ini lebih jauh, kita dapat mengharapkan LLM menyerap pengetahuan tentang perintah tugas sebanyak mungkin selama penyempurnaan. Akibatnya, informasi perintah sebagian ditangkap dalam parameter model, dan LLM yang telah disempurnakan dapat melakukan tugas dengan perintah yang lebih sedikit. Di sini kita mempertimbangkan bentuk perintah sederhana, di mana hanya instruksi (dilambangkan dengan c) dan input pengguna (dilambangkan dengan z) yang disertakan. Perintah dapat dinyatakan menggunakan tupel berikut

Tentukan nilai $x = (c, z)$ (3.11)

Diberikan satu set pasangan prompt-respons $D = \{(x, y)\}$, tujuan fine-tuning adalah untuk meminimalkan total kerugian yang terjadi pada set ini. Metode yang populer adalah meminimalkan log-likelihood negatif (yaitu, memaksimalkan log-likelihood) sehubungan dengan parameter model \hat{y} :

$$\begin{aligned} \hat{y} &= \arg \max_{\hat{y}} \log P(\hat{y}|x) (x, y) \hat{y} D \\ &= \arg \max_{\hat{y}} \log \frac{\text{logaritma } P(\hat{y}|c, z)}{(x, y) \hat{y} D} \end{aligned} \quad (3.12)$$

di mana $P(\hat{y}|\cdot)$ adalah probabilitas yang diprediksi oleh LLM dengan parameter \hat{y} ⁹.

Secara umum, instruksi dalam setiap contoh fine-tuning harus mengikuti pedoman prompt, misalnya, instruksi yang baik harus sejelas mungkin dan memberikan deskripsi tugas secara terperinci. Namun, metode yang dijelaskan dalam persamaan di atas tidak membatasi instruksi ke bentuk tertentu. Fleksibilitas ini memungkinkan kita untuk menginstruksikan LLM dengan cara apa pun yang kita inginkan. Pertimbangkan contoh di mana kita bermaksud menginstruksikan LLM untuk menerjemahkan kalimat bahasa Inggris ke bahasa Mandarin. Tentu saja, seperti yang disebutkan sebelumnya dalam bab ini, kita dapat meminta LLM menggunakan instruksi

Terjemahkan kalimat berikut dari bahasa Inggris ke bahasa Mandarin.

Jika kita ingin instruksinya lebih sederhana, kita dapat mengubahnya menjadi bentuk yang lebih sederhana

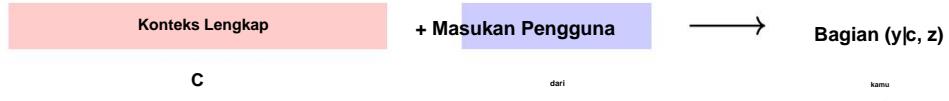
Terjemahkan ini ke dalam bahasa Mandarin.

Bahkan, kita dapat mendefinisikan instruksi sebagai satu frasa

Menerjemahkan!

Dengan upaya penyempurnaan tertentu, kita dapat mengadaptasi LLM untuk mengikuti salah satu instruksi ini. Dari perspektif prompting yang efisien, ada keuntungan komputasi dalam menyederhanakan instruksi dalam prompting. Misalnya, kita dapat menggunakan instruksi sederhana seperti "Terjemahkan!" untuk melakukan tugas yang biasanya memerlukan instruksi yang lebih rumit dan terperinci. Hal ini dapat membuat

⁹Dalam praktiknya, kami menginisialisasi \hat{y} dengan parameter yang diperoleh dari pra-pelatihan, dan kemudian menyesuaikan \hat{y} secara moderat untuk memastikan bahwa hasil setelah penyempurnaan tidak terlalu menyimpang dari hasil yang telah dilatih sebelumnya.

Model Guru:**Model Siswa:**

Gambar 3.4: Ilustrasi distilasi konteks [Snell et al., 2022]. Model guru adalah LLM standar, yang mengambil konteks dan masukan pengguna sebagai masukan model dan menghasilkan prediksi sebagai keluaran model. Kemudian, kami menyederhanakan konteks (misalnya, menyederhanakan instruksi dalam meminta) dan menggunakan model siswa untuk membuat prediksi berdasarkan konteks yang disederhanakan dan masukan pengguna. Model siswa dilatih dengan meminimalkan kerugian antara prediksi yang dihasilkan oleh kedua model.

perintah selama inferensi jauh lebih mudah. Di sisi lain, penyempurnaan LLM dengan instruksi yang terlalu disederhanakan dapat membahayakan generalisasi model. Karena instruksi yang disederhanakan dapat menyebabkan hilangnya informasi, maka LLM cenderung akan menyesuaikan data penyempurnaan secara berlebihan dan gagal untuk menggeneralisasi di luar instruksi tersebut. Dalam skenario yang melibatkan instruksi yang kompleks dan disederhanakan untuk penyempurnaan, masalah ini lebih parah karena data berlabel yang tersedia untuk penyempurnaan biasanya terbatas, dan mengakomodasi berbagai instruksi memerlukan biaya yang mahal.

Cara alternatif untuk mengadaptasi LLM untuk instruksi yang disederhanakan adalah melalui penyulingan pengetahuan. Sebagai contoh, kami mempertimbangkan metode distilasi konteks [Snell et al., 2022]. Tujuan dari metode ini adalah untuk mempelajari model siswa yang dapat menggunakan instruksi yang disederhanakan dari model guru yang mengikuti instruksi dengan baik. Gambar 3.4 menunjukkan ilustrasi dari pendekatan ini. Membangun model guru mengikuti proses penyempurnaan standar: pertama-tama kami mengumpulkan sejumlah data yang mencakup instruksi, masukan pengguna, dan respons yang benar, lalu kami terus melatih model yang telah dilatih sebelumnya dengan kumpulan data ini. Untuk membangun model siswa, kami perlu membangun kumpulan data baru D_y di mana setiap sampel adalah tupel yang terdiri dari instruksi, instruksi sederhana yang sesuai, dan input pengguna, dilambangkan dengan x, z . Distilasi pengetahuan dilakukan dengan meminimalkan fungsi kerugian yang ditentukan pada keluaran model guru dan siswa.

$$\hat{y} = \arg \min_{\hat{y}} \text{Kerugian}(\text{Prt}(\cdot), \text{Prs } \hat{y}(\cdot), x, \cdot) \quad (3.13)$$

di mana $\text{Prt}(\cdot)$ menunjukkan model guru yang telah dilatih sebelumnya, dan $\text{Prs } \hat{y}(\cdot)$ menunjukkan model siswa dengan parameter \hat{y} . Untuk menjaga notasi tetap sederhana, kita akan menulis $\text{Loss}(\text{Prt}(\cdot), \text{Prs } \hat{y}(\cdot), x)$ sebagai Loss untuk singkatnya. Loss yang umum digunakan adalah loss tingkat sekuensial, yang memiliki bentuk dasar:

$$\text{Kerugian} = \text{Prt}(y|c, z) \logaritma \text{Prs } \hat{y}(y|c) \quad (3.14)$$

Namun fungsi ini tidak dapat dihitung secara komputasional karena memerlukan penjumlahan sejumlah output yang sangat besar. Varian dari metode ini adalah melatih model siswa menggunakan output yang dihasilkan oleh model guru. Untuk setiap sampel, kami menggunakan model guru untuk menghasilkan output

$\hat{y} = \arg \max_y \log \text{Pr}_t(y|c, z)$. Kemudian kita pertimbangkan \hat{y} sebagai target pembelajaran, dan fungsi kerugian diberikan oleh

$$\text{Kerugian} = \log \text{Pr}_s(\hat{y}|c) \quad (3.15)$$

Atau, kita dapat meminimalkan jarak antara distribusi probabilitas yang dihasilkan oleh kedua model [Askell et al., 2021]. Misalnya, fungsi kerugian dapat didefinisikan sebagai divergensi KL antara dua distribusi keluaran.

$$\text{Kerugian} = \text{KL}(\text{Pr}_t || \text{Pr}_{\hat{y}}) \quad (3.16)$$

Di mana

$$\text{Pr}_t = \text{Pr}(\cdot | c, z) \quad (3.17)$$

$$\text{Pr}_{\hat{y}} = \text{Pr}_{\hat{s}} \quad (3.18)$$

Meskipun kami telah membatasi diri pada penyulingan pengetahuan untuk instruksi, pendekatan yang dibahas di sini bersifat umum. Dengan belajar dari keluaran model guru, pengetahuan dalam mendorong dapat disulung menjadi parameter model siswa. Oleh karena itu, model yang disulung dapat dianggap sebagai pengkodean semacam perintah lunak. Metode ini dapat diterapkan pada banyak masalah lain dalam pembelajaran perintah lunak, seperti mengompresi konteks yang panjang dan mempelajari perintah lunak sebagai komponen khusus LLM.

3.3.2.2 Mempelajari Soft Prompt untuk Fine-tuning yang Efisien terhadap Parameter

Memperbarui semua parameter merupakan metode umum untuk mengadaptasi LLM ke tugas yang diminati. Meskipun fine-tuning dianggap lebih murah secara komputasi daripada pra-pelatihan, namun tetap mahal untuk diterapkan dalam praktik. Masalah ini memotivasi pengembangan metode fine-tuning yang hemat parameter, yang bertujuan untuk meminimalkan jumlah parameter yang perlu diperbarui.

Salah satu pendekatan, yang dikenal sebagai penyetelan awalan, adalah menambahkan serangkaian vektor yang dapat dilatih, atau awalan, di awal masukan setiap lapisan Transformer [Li dan Liang, 2021]. Awalan ini dapat dianggap sebagai perintah lunak yang berfungsi sebagai konteks tambahan untuk memandu perilaku model dalam tugas tertentu. Selama penyetelan awalan, kita hanya perlu mempelajari awalan untuk menanamkan pengetahuan khusus tugas. Dengan demikian, metode ini efisien karena hanya memodifikasi sebagian kecil model daripada menyesuaikan seluruh rangkaian parameter model.

Secara khusus, biarkan masukan lapisan pada kedalaman / dilambangkan dengan $H_l = h_1^l \dots h_M^l$. Hasil dari $0h$ lapisan tersebut dapat dinyatakan sebagai

$$H_{l+1} = \text{Lapisan}(H_l) \quad (3.19)$$

Dalam penyetelan awalan, kami memperluas urutan $0h = h_1^0 \dots h_M^0$ dengan menambahkan beberapa vektor di awal, yang kami sebutkan sebagai $p^{0, 1, \dots, p_N}$. Oleh karena itu H_l dapat ditulis dalam bentuk

$$H_l = p \underbrace{\dots}_{\text{dapat dilatih}} \underbrace{h_1^0 \dots h_M^0}_{\text{keluaran lapisan sebelumnya}} \quad (3.20)$$

Output dari lapisan tersebut adalah representasi $m + 1$ terakhir.

$$\overline{H}^{aku+1} = \text{Lapisan}(Hl)[\underline{jym \dots y1}] = h_{l+1} \\ h_{l+1} \dots h_{l+1} \quad M \quad (3.21)$$

di mana $\underline{jym \dots y1}$ menyatakan operasi pemotongan yang mengekstrak $m + 1$ elemen terakhir dari suatu barisan.

Diberikan H_{l+1} , masukan lapisan berikutnya dapat dinyatakan dalam bentuk yang sama dengan Persamaan (3.20):

$$H_{l+1} = p \quad \begin{matrix} \overline{aku+1} & \overline{aku+1} & \overline{aku+1} & H_0 & p & \dots & p_{l-1} \\ \overline{aku+1} & \overline{aku+1} & = p_0 & p & N & & \end{matrix} \quad \overline{aku+1} \\ 1 \quad \overline{aku+1} \quad \overline{aku+1} \quad \overline{aku+1} \quad h_{l+1} \quad h_{l+1} \dots h_{l+1} \quad (3.22)$$

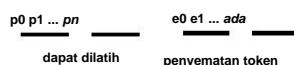
Di sini setiap $p_i \in R^D$ dapat dilihat sebagai parameter yang dapat dipelajari. Selama pelatihan $p_0 \dots p_l$ sudah terlatih seperti biasa, dan parameter model Transformer asli tetap sama.

Gambar 3.5 menunjukkan ilustrasi penyetelan halus prefiks untuk tugas penerjemahan. Di sini, hanya prefiks yang diperbarui dengan menerima input tugas $\underline{\quad}$ dan p_{l-1} gradien kesalahan dari keluaran (yaitu, vektor terjemahan p dalam bahasa Mandarin). Dengan menyesuaikan vektor-vektor ini penerjemahan, model beradaptasi sebagaimana mestinya. Hal ini menjadikan p dan p_{l-1} berfungsi sebagai perintah yang mengaktifkan LLM untuk melakukan tugas tanpa memerlukan perintah masukan eksplisit seperti "Terjemahkan kalimat berikut dari bahasa Inggris ke bahasa Mandarin". Pada waktu pengujian, kami menambahkan p yang dioptimalkan ke lapisan, dan LLM kemudian akan menerjemahkan kalimat masukan 0. Perhatikan bahwa penyetelan halus prefiks memperkenalkan parameter $L \times n \times d$ tambahan, di mana L adalah jumlah lapisan, n adalah jumlah prefiks, dan d adalah dimensionalitas setiap prefiks. Namun, jumlah ini jauh lebih kecil dibandingkan dengan jumlah total parameter dalam LLM, sehingga proses penyetelan halus menjadi sangat efisien.

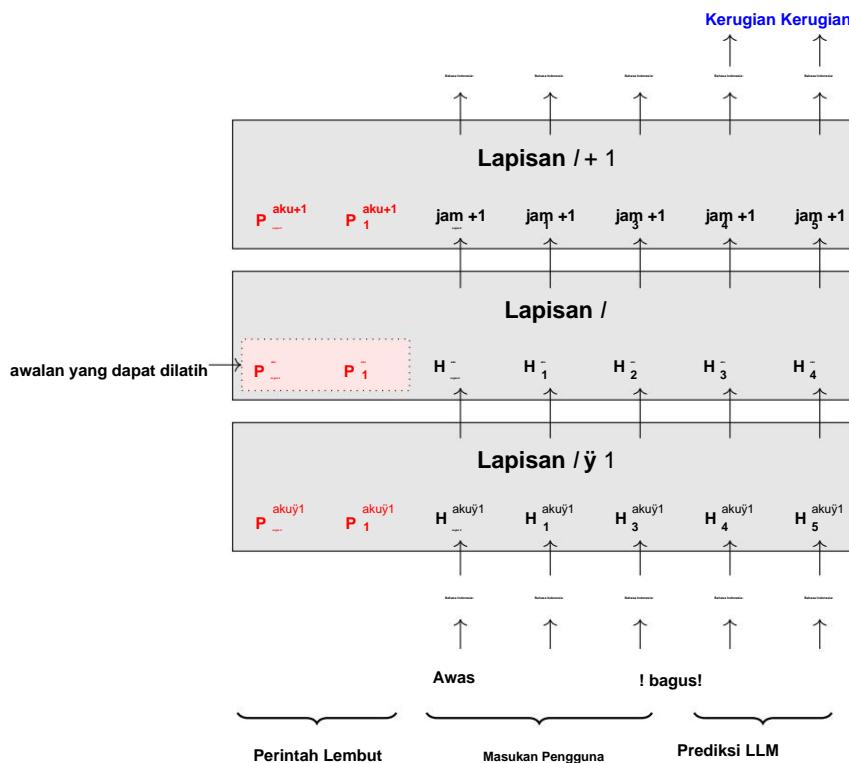
Meskipun penyempurnaan awalan itu mudah, hal itu tetap memerlukan modifikasi pada LLM. Atau, memisahkan perintah lunak dari LLM memungkinkan kita untuk mempertahankan arsitektur model asli, sehingga lebih efisien untuk penerapan di berbagai tugas tanpa perlu menyesuaikan model inti.

Salah satu metode tersebut adalah prompt tuning [Lester et al., 2021]. Seperti halnya prefix fine-tuning, prompt tuning menggabungkan vektor yang dapat dilatih sehingga LLM dapat beradaptasi dengan tugas yang diberikan dengan menyesuaikan vektor tersebut. Namun, penyetelan cepat berbeda karena hanya memodifikasi lapisan penyematan.

Ingatlah bahwa dalam LLM, setiap token masukan z_i direpresentasikan oleh e_i yang disematkan. Penyematan ini umumnya dipelajari melalui model penyematan token dan kemudian digunakan sebagai input nyata ke LLM, mengantikan token yang direpresentasikan secara simbolis. Dalam penyetelan cepat, sejumlah penyematan semu $p_0 \dots p_n$ ditambahkan di awal urutan penyematan token. Jadi input aktual ke LLM dapat dinyatakan sebagai



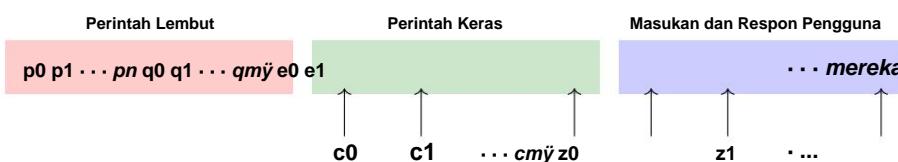
Perhatikan bahwa penyematan semu tidak perlu sesuai dengan token apa pun dalam bahasa alami. Sebaliknya, penyematan ini dapat dilihat sebagai "penyematan perintah lunak" yang berfungsi untuk mengondisikan LLM. Dengan melatih penyematan perintah lunak pada data khusus tugas, mereka belajar berinteraksi secara adaptif dengan penyematan token $e_0 \dots e_m$ dan memandu perilaku LLM. Karena penyetelan perintah tidak mengubah parameter dasar LLM yang telah dilatih sebelumnya, ini dianggap sebagai metode penyetelan yang ringan dan efisien, yang meningkatkan kinerja khusus tugas sambil mempertahankan kemampuan generalisasinya. Lihat Gambar 3.6 untuk ilustrasi penyetelan perintah.



Gambar 3.5: Ilustrasi penyempurnaan awalan untuk tugas penerjemahan (Awas! ý ýí!). Untuk setiap lapisan, kami menambahkan dua awalan p_- dan p_1 di awal. LLM dilatih untuk meminimalkan kerugian pada prediksi yang diberikan input.

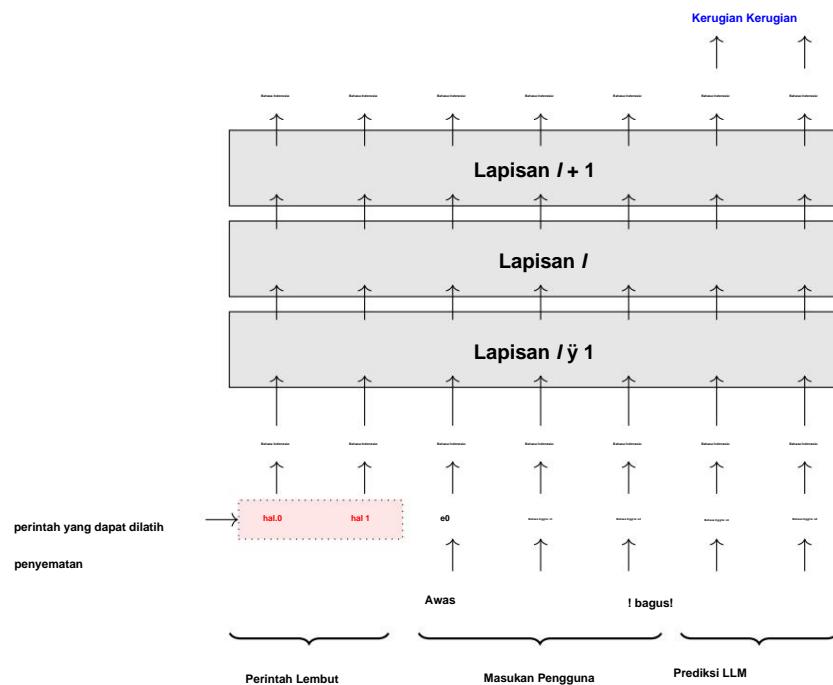
Selama proses ini, hanya awalan yang dioptimalkan sementara parameter lainnya tetap. Oleh karena itu, model dapat beradaptasi dengan tugas yang diberikan dengan cara yang sangat efisien. Pada waktu inferensi, LLM bekerja dengan awalan yang dioptimalkan, dan dapat melaksanakan tugas tanpa memerlukan perintah keras yang eksplisit.

Karena $p_0 p_1 \dots p_n$ sendiri merupakan suatu barisan, kita dapat menggunakan model barisan untuk merepresentasikannya dengan lebih baik. Misalnya, model Transformer dapat mengkodekan urutan ini, dan representasi yang dihasilkan kemudian dapat digunakan sebagai input untuk LLM. Dengan kata lain, kita dapat mengembangkan model tambahan untuk mengodekan perintah lunak. Cara lain untuk meningkatkan perintah adalah dengan menggabungkan perintah lunak dan perintah keras. prompt, sehingga memanfaatkan kedua jenis [Liu et al., 2023b]. Dalam urutan penyematan, kita dapat mengatur atau menyelingi prompt ini. Ini akan menghasilkan pola prompt yang berbeda. Misalnya misalnya pola sederhana yang menggunakan kedua jenis prompt adalah



di mana $c_0 \dots c_m y$ menunjukkan prompt keras dan $q_0 \dots q_m y$ menunjukkan urutan penyematan yang sesuai. urutan.

Di sini kami telah mempertimbangkan metode untuk memasukkan soft prompt di LLM. Namun kami melewatkannya detailnya pelatihan prompt lunak ini dan berasumsi bahwa pembaca sudah familiar dengan standar yang diawasi proses pembelajaran, yaitu memaksimalkan kemungkinan keluaran model yang benar berdasarkan model tersebut



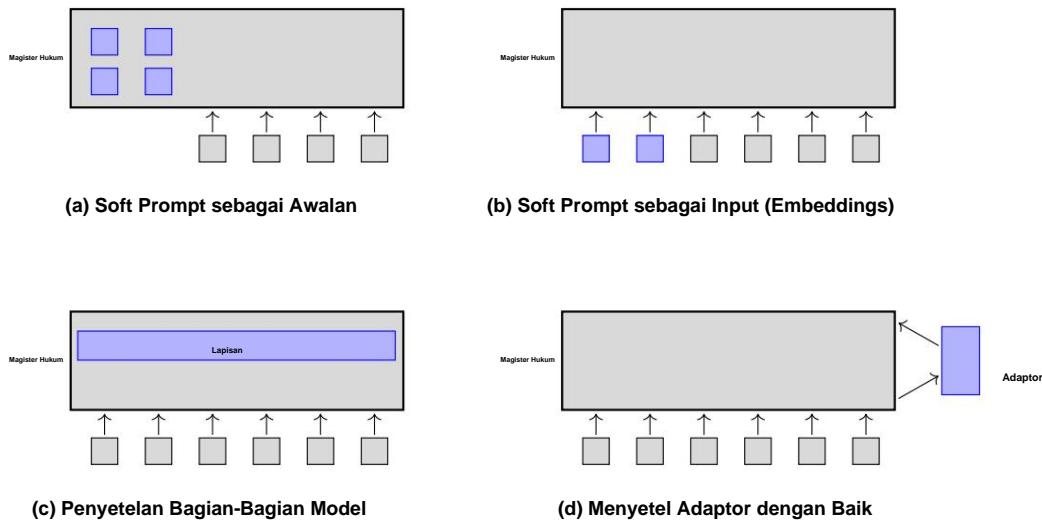
Gambar 3.6: Ilustrasi penyetelan prompt untuk tugas penerjemahan (Awas! ý ý!). Alih-alih menggunakan teks tetap prompt, prompt lunak adalah penyematan yang dapat dipelajari yang ditambahkan di awal urutan penyematan. Selama fine-tuning, hanya penempatan perintah ini yang dioptimalkan untuk secara efisien mengadaptasi LLM ke tugas yang diberikan. Setelah dioptimalkan, penempatan perintah digunakan untuk menginstruksikan LLM untuk melakukan tugas saat data baru masuk.

input. Faktanya, mempelajari soft prompt dapat dikaitkan dengan banyak masalah dalam penyempurnaan LLM. Misalnya, jika kita menganggapnya sebagai masalah kompresi konteks, kita dapat menerapkan penyulingan pengetahuan metode yang dijelaskan sebelumnya. Dalam karya Mu et al. [2024], prompt dikompresi dan direpresentasikan sebagai beberapa token semu, yang ditambahkan ke setiap urutan input. Penyematan token semu ini dioptimalkan untuk meniru prediksi model yang diminta standar. Dengan kata lain, pengetahuan pemicu disulung dari model guru menjadi token semu.

Secara umum, banyak metode fine-tuning yang efisien parameter dapat dianggap sebagai pembelajaran semacam perintah lunak [Lialin et al., 2023]. Ketika kita menyempurnakan bagian dari LLM untuk suatu tugas, ini Proses ini pada dasarnya dapat dilihat sebagai penyuntikan informasi perintah terkait tugas ke dalam tugas tertentu. bagian dari model. Pendekatan lain yang banyak digunakan untuk fine-tuning parameter-efisien adalah dengan menambahkan lapisan adaptor antara lapisan model yang ada. Pendekatan ini memungkinkan kita untuk menyempurnakan hanya lapisan adaptor pada tugas-tugas tertentu tanpa mengubah arsitektur yang mendasarinya atau melatih ulang keseluruhannya model. Dalam hal ini, lapisan adaptor dapat dilihat sebagai perintah lunak yang mengodekan perintah dan informasi terkait tugas dan berinteraksi dengan LLM asli untuk membantunya beradaptasi. Sebagai rangkuman, Gambar 3.7 menunjukkan perbandingan berbagai metode penggunaan soft prompt di LLM.

3.3.2.3 Mempelajari Soft Prompt dengan Kompresi

Pendekatan lain untuk mempelajari soft prompt adalah dari perspektif kompresi. Sebagai contoh sederhana misalnya, pertimbangkan masalah pendekatan konteks panjang menggunakan representasi berkelanjutan [Wingate et al., 2022]. Misalkan kita memiliki input pengguna z dan konteksnya c (seperti instruksi panjang dan demonstrasi). Sekarang kita ingin mengembangkan representasi konteks yang terkompresi, yang dilambangkan



Gbr. 3.7: Ilustrasi penggunaan soft prompt dalam LLM. Di sini, soft prompt yang dapat disetel ditunjukkan dengan warna biru, dan komponen yang parameternya ditetapkan selama penyempurnaan ditunjukkan dengan warna abu-abu. Pada subgambar (a), soft prompt adalah awalan yang ditambahkan ke setiap lapisan LLM. Pada subgambar (b), soft prompt digunakan sebagai penyematan masukan untuk LLM. Pada subgambar (c) dan (d), soft prompt secara umum diperlukan sebagai komponen model yang disetel dengan baik untuk adaptasi tugas.

oleh \hat{y} , sehingga prediksi berdasarkan z dan \hat{y} sedekat mungkin dengan prediksi berdasarkan z dan c . Sasaran ini dapat dinyatakan dalam bentuk

$$\hat{y} = \arg \min_{\hat{y}} s(\hat{y}, y|\hat{y}) \quad (3.23)$$

di mana $\hat{y} = \arg \max_y \Pr(y|c, z)$ dan $y|\hat{y} = \arg \max_{\hat{y}} \Pr(y|\hat{y}, z)$ masing-masing adalah prediksi LLM yang diberikan konteks penuh dan konteks terkompresi. Fungsi $s(\cdot, \cdot)$ biasanya merepresentasikan ukuran kehilangan atau kesamaan, yang bertujuan untuk meminimalkan perbedaan prediksi antara dua representasi konteks.

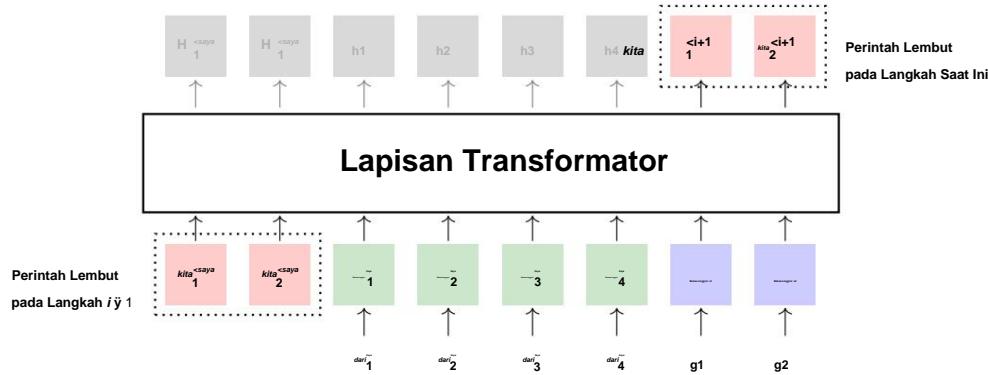
Salah satu kerangka kerja umum untuk mencapai hal ini adalah penyulingan pengetahuan, di mana \hat{y} dan $y|\hat{y}$ dapat dilihat sebagai prediksi model guru dan model siswa, masing-masing. Formalisasi ini menghubungkan pembahasan kita dengan masalah penyulingan konteks yang dibahas sebelumnya. Sasaran pelatihan dapat diperoleh dengan analogi dengan Persamaan (3.15) dan (3.16). Misalnya, sasaran pelatihan sederhana diberikan oleh

$$\hat{y} = \arg \max_{\hat{y}} \log \Pr(\hat{y}|y|\hat{y}, z) \quad (3.24)$$

Sebagai alternatif, kita dapat meminimalkan divergensi KL antara distribusi output, memberikan

$$\arg \min_{\hat{y}} \text{KL}(\Pr(\cdot|c, z) \parallel \Pr(\cdot|\hat{y}, z)) \hat{y} = \quad (3.25)$$

Perbedaannya dengan model dalam Persamaan (3.15) dan (3.16) adalah bahwa di sini konteks terkompresi direpresentasikan sebagai vektor bernilai riil (sebut saja prompt embedding), dan bukan sebagai token normal. Dengan menerapkan metode di atas, kita menyaring konteks dari urutan token c ke dalam penyisipan \hat{y} . Perhatikan bahwa model guru $\Pr(\cdot|c, z)$ dan model siswa $\Pr(\cdot|\hat{y}, z)$ mungkin tidak berbagi arsitektur atau pengaturan model yang sama. Dalam praktiknya, kita umumnya menginginkan model guru menjadi



Gambar 3.8: Ilustrasi kompresi segmen konteks menjadi perintah lunak ($\hat{y} = 2$ dan $mi = 4$). Input ke perintah LLM mencakup perintah lunak dari langkah sebelumnya ($\hat{y}_1^{<\text{saya}}$ dan $\hat{y}_2^{<\text{saya}}$), token segmen (z_1, z_2, z_3 , dan z_4), dan token ringkasan (g_1 dan g_2). Mengingat hal ini, LLM beroperasi seperti biasa. Kami kemudian mengekstrak output di lapisan Transformer terakhir yang sesuai dengan token ringkasan. Output ini dapat dilihat sebagai perintah lunak yang terakumulasi hingga segmen ini.

lebih kuat, sementara model siswa harus lebih kecil dan lebih efisien.

Meskipun mengompresi konteks penuh menjadi representasi berkelanjutan adalah pendekatan yang mudah untuk mempelajari soft prompt, diperlukan model guru yang dapat menangani rangkaian input yang panjang. Namun, dalam banyak kasus, konteksnya begitu panjang sehingga penerapan LLM terlalu mahal atau tidak layak. Pemodelan urutan input yang panjang dapat termasuk dalam keluarga besar metode yang efisien untuk LLM konteks panjang. Banyak teknik telah dikembangkan untuk mengatasi masalah ini. Misalnya, seseorang dapat menggunakan cache KV berukuran tetap untuk menyimpan informasi masa lalu di setiap langkah selama inferensi. Efisiensi arsitektur transformator dan LLM konteks panjang telah dibahas secara intensif dalam buku ini. Untuk pembahasan lebih rinci mengenai topik ini, pembaca yang tertarik dapat merujuk ke Bab 2.

Ada juga metode yang dirancang khusus untuk mengompresi konteks panjang menjadi perintah lunak. Berikut ini Kami mempertimbangkan metode Chevalier et al. [2023] sebagai contoh. Ide dasarnya adalah kita belajar prompt lunak secara bertahap dengan mengumpulkan representasi konteks ukuran tetap atas konteks urutan. Mengingat konteks yang panjang, pertama-tama kita membaginya menjadi sejumlah segmen z^1, \dots, z^K . Kami kemudian memproses segmen-segmen ini secara berurutan, setiap kali menghasilkan representasi konteks yang kita inginkan. telah diproses sejauh ini, dilambangkan dengan $\hat{y}^{<i+1}$. Untuk melakukan ini, beberapa token ringkasan $g_1, \dots, g_{\hat{y}}$ adalah diperkenalkan. Pada setiap langkah, kami mengambil segmen $z^{<\text{saya}}_1, \dots, z^{<\text{saya}}_{\hat{y}}$, bersama dengan konteks sebelumnya $\text{rep-}^{<i}$ penolakan $\hat{y}^{<i}$ dan token ringkasan $g_1, \dots, g_{\hat{y}}$ sebagai input, dan menggunakan LLM untuk menghasilkan urutan representasi tersembunyi yang sesuai pada lapisan Transformer terakhir. Contoh dari ini Prosesnya diilustrasikan pada Gambar 3.8.

Di sini $\hat{y}^{<i}$ pada dasarnya adalah memori. Model ini beroperasi dalam mode RNN. Setiap kali kita mengambil segmen dan memperbarui memori ini dengan mengodekan status memori sebelumnya dan segmen. Oleh karena itu, $\hat{y}^{<i}$ yang dihasilkan pada segmen terakhir merupakan representasi dari keseluruhan rangkaian konteks. Model Transformer untuk mempelajari representasi ini bisa menjadi LLM standar tetapi kita perlu menyempurnakannya agar beradaptasi dengan tugas representasi konteks ini.

Perlu dicatat bahwa di sini kita hanya menganggap prompt dan konteks sebagai istilah yang serupa, meskipun keduanya tidak sama. Meskipun kita agak “menyalahgunakan” konsep prompt, kita sering melihatnya sebagai suatu jenis konteks. Dari perspektif ini, metode yang dibahas di sini dapat diterapkan pada teks umum masalah kompresi.

3.3.3 Pengurangan Panjang Prompt

Meskipun perintah lunak menyediakan representasi yang padat dan tersembunyi, perintah tersebut tidak dapat diinterpretasikan secara langsung. Kurangnya kemampuan interpretabilitas dapat menjadi hambatan yang signifikan bagi pengguna yang mencoba memahami bagaimana masukan mereka memengaruhi keluaran LLM. Selain itu, meskipun perintah lunak efisien untuk penyempurnaan dan penerapan, perintah tersebut tidak fleksibel dan tidak memungkinkan penyesuaian yang mudah tanpa penyempurnaan atau modifikasi yang ekstensif. Ketidakfleksibelan ini dapat membatasi kegunaannya dalam lingkungan yang dinamis di mana perubahan perintah sering kali diperlukan.

Salah satu cara alternatif untuk mengembangkan perintah yang efisien adalah dengan menyederhanakan teks yang digunakan untuk perintah tersebut. Misalnya, di bawah ini adalah perintah untuk menjawab pertanyaan tentang perawatan kesehatan dan keuangan.

Tugasnya meliputi pengembangan model bahasa yang mampu memahami dan menanggapi pertanyaan pengguna di berbagai domain, dengan penekanan khusus pada perawatan kesehatan dan keuangan. Dengan mempertimbangkan berbagai pertanyaan potensial, mulai dari diagnosis medis hingga nuansa regulasi keuangan, model tersebut harus memastikan pemahaman yang komprehensif dan tanggapan yang akurat.

Pertanyaan:

Apa praktik terbaik untuk menggunakan kecerdasan buatan dalam mendiagnosis penyakit kardiovaskular?

Kita dapat menyederhanakan deskripsi tugas dengan menghapus bagian-bagian yang tidak penting.

Tugasnya meliputi pengembangan model bahasa yang mampu memahami dan menanggapi pertanyaan pengguna di berbagai domain, dengan penekanan khusus pada perawatan kesehatan dan keuangan. Dengan mempertimbangkan berbagai pertanyaan potensial, mulai dari diagnosis medis hingga nuansa regulasi keuangan, model tersebut harus memastikan pemahaman yang komprehensif dan tanggapan yang akurat.

Kita juga dapat memparafrasekannya menjadi teks yang lebih pendek.

Tugasnya melibatkan pengembangan model bahasa yang berfokus pada perawatan kesehatan dan keuangan, yang mampu memahami dan menanggapi secara akurat berbagai pertanyaan pengguna.

Masalah ini dapat dilihat sebagai masalah NLP klasik — penyederhanaan teks. Jadi metode yang digunakan dapat bersifat umum dan tidak terbatas pada masalah penyederhanaan perintah. Ada banyak cara untuk mencapainya. Salah satu metode sederhana adalah dengan menentukan beberapa heuristik dan mengidentifikasi kata-kata yang berlebihan yang dapat dihilangkan tanpa kehilangan informasi penting. Misalnya, kita dapat memeriksa setiap token dalam suatu urutan dalam hal kontribusinya terhadap makna keseluruhan dan menghapus yang memberikan nilai minimal [Li et al., 2023c; Jiang et al., 2023b]. Metode lain melibatkan pembingkaiyan masalah sebagai tugas urutan ke urutan. Dengan data berlabel untuk penyederhanaan teks, kita dapat melatih model encoder-decoder untuk mengubah setiap teks masukan menjadi bentuk yang disederhanakan. Selain itu,

mengingat banyak LLM telah disempurnakan dan diselaraskan untuk menjalankan tugas penyederhanaan teks, mudah untuk menggunakan model ini guna menyederhanakan perintah. Misalnya, kita dapat meminta LLM untuk menyederhanakan teks dengan batasan tertentu, seperti membatasi panjang teks yang disederhanakan.

3.4 Ringkasan

Dalam bab ini, kami telah membahas berbagai masalah yang terkait dengan dorongan LLM. Pembahasan kami terutama difokuskan pada dua aspek:

- Bagaimana merancang petunjuk dasar untuk memandu prediksi LLM dan menyempurnakan petunjuk tersebut untuk pemecahan masalah yang lebih efektif dan efisien?
- Bagaimana cara mengotomatiskan desain dan representasi perintah?

Solusi untuk masalah ini melibatkan desain perintah umum dan teknik yang lebih maju, seperti CoT dan pembelajaran perintah, yang telah dieksplorasi secara ekstensif dalam penelitian terkini.

Dalam NLP, prompting dapat dilihat sebagai teknologi yang telah berkembang seiring dengan LLM, dan dalam arti tertentu, prompting telah membuka pintu bagi penerapan praktis model-model ini dalam berbagai domain masalah yang mengesankan. Bahkan, jika kita memperluas konsep prompt sampai batas tertentu, prompting dapat ditelusuri kembali ke masa-masa awal pembelajaran mesin dan NLP. Misalnya, banyak sistem NLP menggunakan fitur dan templat yang dibuat sendiri untuk "meminta" tugas-tugas tertentu. Bayangkan mengembangkan fitur untuk menunjukkan apakah suatu teks bersifat formal atau informal. Kita dapat memasukkan fitur ini ke dalam sistem untuk mengkondisikan terjemahan pada jenis teks masukan.

Penggunaan konsep prompt modern secara luas dimulai dengan munculnya model pra-latihan besar di bidang NLP. Awalnya, model-model ini, seperti BERT, diadaptasi untuk tugas-tugas hilir tertentu terutama melalui penyempurnaan. Namun, para peneliti segera menemukan bahwa dengan merancang "prompt" tertentu — menambahkan kata-kata atau kalimat tertentu ke input — model-model tersebut dapat dipicu untuk merespons tugas-tugas tertentu tanpa penyempurnaan yang ekstensif. Hal ini memotivasi komunitas NLP untuk mengembangkan dan menerapkan model-model dasar universal yang dapat dipicu untuk menangani berbagai tugas tanpa mengubah arsitektur dasar dan prosedur pra-pelatihan.

Pendekatan prompting pertama kali diujicobakan dengan model yang lebih kecil dan kemudian menunjukkan kemampuan yang mengesankan dengan model besar seperti GPT-3, yang dapat menghasilkan teks berkualitas tinggi sebagai respons terhadap prompt sederhana di berbagai tugas. Seiring berkembangnya teknologi prompting, rekayasa prompt muncul sebagai area penelitian yang penting. Seperti yang dibahas dalam bab ini, secara luas melibatkan perancangan prompt yang efektif untuk memaksimalkan kinerja model, yang mencakup prompt yang dibuat dengan tangan dan yang dibuat secara otomatis. Penelitian yang lebih baru telah mengeksplorasi cara meningkatkan efektivitas prompting melalui teknik seperti few-shot learning, zero-shot learning, dan CoT reasoning, yang memungkinkan LLM bekerja secara efektif di berbagai skenario. Pembahasan umum tentang prompting bisa sangat luas, dan kami tidak dapat membahas semua detail dalam bab ini. Untuk teknik pemberian petunjuk yang lebih maju, pembaca dapat merujuk ke survei terkini. Topiknya meliputi pembelajaran dalam konteks [Li, 2023; Dong et al., 2022], CoT [Chu et al., 2023; Yu et al., 2023; Zhang et al., 2023a], pemberian petunjuk yang efisien [Chang et al., 2024], dan rekayasa petunjuk umum [Liu et al., 2023c; Chen et al., 2023a].

Perhatikan bahwa meskipun idealnya kita ingin mengembangkan metode prompting umum tanpa menyesuaikan arsitektur dan parameter model, hasil prompting umumnya sangat bergantung pada kualitas dan ukuran LLM yang diberikan. Untuk model yang lebih kuat, seperti LLM daring yang dikomersialkan, prompt sederhana mungkin cukup untuk menginstruksikan model ini untuk melakukan tugas dengan benar. Dalam kasus ini, rekayasa prompt relatif mudah, meskipun kita masih memerlukan upaya tertentu untuk membuat LLM berfungsi dengan baik. Sebaliknya, jika LLM tidak cukup kuat, kita mungkin perlu merancang prompt dengan hati-hati untuk mencapai hasil yang diinginkan. Dalam banyak kasus, penyempurnaan masih diperlukan untuk mengadaptasi model ke strategi prompting yang

BAB 4

Penyelarasan

Penyelarasan bukanlah konsep baru dalam NLP, tetapi maknanya bervariasi di berbagai domain dan seiring waktu. Dalam NLP tradisional, istilah penyelarasan biasanya merujuk pada tugas yang menghubungkan elemen yang sesuai dalam dua set, seperti menyelaraskan kata-kata antara kalimat bahasa Mandarin dan kalimat bahasa Inggris. Karena LLM menjadi semakin penting dalam penelitian NLP, istilah ini lebih luas digunakan untuk merujuk pada penyelarasan keluaran model dengan harapan manusia. Masalah yang diatasi oleh penyelarasan adalah bahwa keluaran suatu model mungkin tidak selaras dengan tujuan atau konteks spesifik yang diinginkan oleh pengguna. Misalnya, LLM yang telah dilatih sebelumnya mungkin tidak dapat mengikuti petunjuk pengguna karena tidak dilatih untuk melakukannya. Contoh lain adalah bahwa LLM dapat menghasilkan konten yang berbahaya atau melestarikan bias yang melekat dalam data pelatihannya. Hal ini menimbulkan tantangan baru dalam memastikan bahwa keluaran LLM tidak hanya akurat dan relevan, tetapi juga etis dan tidak diskriminatif.

Sekadar melakukan pra-pelatihan LLM dapat mengakibatkan berbagai masalah penyelarasan. Sasaran akhir kami adalah untuk mengatasi atau mengurangi semua masalah ini guna memastikan LLM akurat dan aman. Ada masalah menarik di sini: karena model bahasa yang besar dilatih pada sejumlah besar data, kami punya alasan untuk percaya bahwa jika kami memiliki data yang cukup yang mencakup berbagai tugas dan selaras dengan preferensi manusia, pra-pelatihan dapat membuat LLM cukup akurat dan aman, bahkan mungkin menghilangkan kebutuhan untuk penyelarasan. Namun, kenyataannya adalah bahwa hampir mustahil untuk mengumpulkan data yang mencakup semua tugas atau cukup mewakili preferensi manusia. Hal ini membuat sulit untuk mencapai penyelarasan model melalui pra-pelatihan saja, atau setidaknya, pada tahap ini, penyelarasan tetap menjadi langkah yang sangat diperlukan dan penting dalam pengembangan LLM.

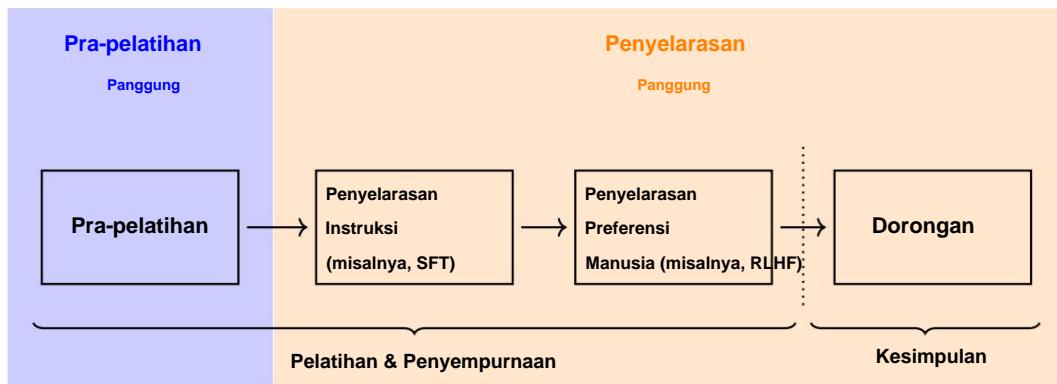
Dalam bab ini, kami akan fokus pada metode penyelarasan untuk LLM. Kami akan mulai dengan membahas tugas penyelarasan umum. Kemudian kami akan mempertimbangkan dua pendekatan yang umum digunakan, yang dikenal sebagai penyelarasan instruksi dan penyelarasan preferensi manusia. Yang pertama menggunakan teknik fine-tuning yang diawasi dan memandu LLM untuk menghasilkan keluaran yang mematuhi instruksi pengguna. Di sisi lain, yang terakhir biasanya bergantung pada teknik pembelajaran penguatan, di mana LLM dilatih berdasarkan umpan balik dari manusia. Meskipun metode ini dimotivasi oleh tujuan yang berbeda, metode ini umumnya digunakan bersama untuk mengembangkan LLM yang selaras dengan baik.

4.1 Tinjauan Umum Penyelarasan LLM

Penyelarasan dapat dicapai dengan beberapa cara berbeda. Kita memerlukan metode yang berbeda untuk penyelarasan LLM karena masalah ini sendiri rumit dan memiliki banyak sisi, yang memerlukan perpaduan berbagai pertimbangan teknis. Di sini kita mempertimbangkan tiga pendekatan yang umum digunakan untuk menyelaraskan LLM.

Pendekatan pertama adalah menyempurnakan LLM dengan data berlabel. Pendekatan ini mudah karena hanya memperluas pelatihan LLM yang telah dilatih sebelumnya untuk menyesuaikannya dengan tugas-tugas tertentu. Salah satu contohnya adalah penyempurnaan terbimbing (supervised fine-tuning/SFT), di mana LLM dilatih lebih lanjut pada kumpulan data yang terdiri dari instruksi khusus tugas yang dipasangkan dengan keluaran yang diharapkan. Kumpulan data SFT umumnya jauh lebih kecil dibandingkan dengan kumpulan pelatihan asli, tetapi data ini sangat terspesialisasi. Hasil dari SFT adalah LLM dapat belajar untuk menjalankan tugas berdasarkan instruksi pengguna.

Misalnya, dengan menyempurnakan LLM dengan serangkaian pasangan pertanyaan-jawaban, model dapat menjawab pertanyaan-pertanyaan tertentu, bahkan jika tidak secara langsung tercakup dalam dataset SFT. Metode ini terbukti sangat efektif.



Gbr. 4.1: Ilustrasi skematis metode pra-latih-lalu-sejajarkan untuk mengembangkan LLM. Pada tahap pra-pelatihan, kami melatih LLM pada sejumlah besar data menggunakan prediksi token berikutnya. Kemudian, pada tahap penyelarasan, kami menyelaraskan LLM dengan instruksi, maksud, dan preferensi pengguna. Ini termasuk penyelarasan instruksi, penyelarasan preferensi manusia, dan perintah.

berguna ketika relatif mudah untuk menggambarkan hubungan input-output dan mudah untuk membuat anotasi pada data.

Pendekatan kedua adalah menyempurnakan LLM menggunakan model penghargaan. Salah satu kesulitan dalam penyelarasan adalah nilai-nilai dan harapan manusia itu rumit dan sulit dijelaskan. Dalam banyak kasus, bahkan bagi manusia sendiri, mengartikulasikan apa yang secara etis benar atau sesuai secara budaya dapat menjadi tantangan. Akibatnya, pengumpulan atau pencatatan data penyempurnaan tidak semudah dengan SFT. Selain itu, menyelaraskan LLM bukan sekadar tugas untuk menyesuaikan data, atau dengan kata lain, sampel terbatas yang diberi anotasi oleh manusia sering kali tidak cukup untuk menggambarkan perilaku ini secara komprehensif. Yang benar-benar kita perlukan di sini adalah mengajarkan model cara menentukan keluaran mana yang lebih sesuai dengan preferensi manusia, misalnya, kita tidak hanya ingin keluaran tersebut akurat secara teknis tetapi juga selaras dengan harapan dan nilai manusia. Salah satu idenya adalah mengembangkan model penghargaan yang dianalogikan dengan pakar manusia. Model penghargaan ini akan bekerja dengan memberi penghargaan kepada LLM setiap kali menghasilkan respons yang lebih sesuai dengan preferensi manusia, seperti cara guru memberikan umpan balik kepada siswa. Untuk memperoleh model penghargaan seperti itu, kita dapat melatih fungsi penilaian dari data preferensi manusia. Model penghargaan yang dilatih kemudian digunakan sebagai panduan untuk menyesuaikan dan menyempurnakan LLM. Ini membungkai tugas penyelarasan LLM sebagai tugas pembelajaran penguatan. Metode yang dihasilkan, seperti pembelajaran penguatan dari umpan balik manusia (RLHF), telah terbukti sangat berhasil dalam mengadaptasi LLM untuk mengikuti

Pendekatan ketiga adalah melakukan penyelarasan selama inferensi, bukan selama pelatihan atau penyempurnaan. Dari perspektif ini, pemberian petunjuk dalam LLM juga dapat dilihat sebagai bentuk penyelarasan, tetapi tidak melibatkan pelatihan atau penyempurnaan. Jadi, kita dapat secara dinamis mengadaptasi LLM ke berbagai tugas dengan biaya minimal. Metode lain untuk melakukan penyelarasan pada waktu inferensi adalah dengan menilai ulang keluaran LLM. Misalnya, kita dapat mengembangkan sistem penilaian untuk mensimulasikan umpan balik manusia pada keluaran LLM (seperti model penghargaan) dan memprioritaskan

Ketiga metode yang disebutkan di atas biasanya digunakan secara berurutan setelah pra-pelatihan selesai: pertama-tama kami melakukan SFT, kemudian RLHF, dan kemudian meminta LLM dengan cara tertentu selama inferensi. Ini secara kasar membagi pengembangan LLM menjadi dua tahap — tahap pra-pelatihan dan tahap penyelarasan. Gambar 4.1 menunjukkan ilustrasinya. Karena teknik permintaan telah dibahas secara intensif dalam bab sebelumnya, kami akan fokus pada metode penyelarasan berbasis fine-tuning di sisa bab ini.

4.2 Penyelarasan Instruksi

Salah satu fitur LLM adalah mereka dapat mengikuti perintah yang diberikan oleh pengguna untuk melakukan berbagai tugas. Dalam banyak aplikasi, prompt terdiri dari instruksi sederhana dan input pengguna, dan kami ingin LLM untuk mengikuti instruksi ini untuk melakukan tugas dengan benar. Kemampuan LLM ini juga disebut kemampuan mengikuti instruksi. Misalnya, di bawah ini adalah perintah di mana kita ingin LLM untuk mengekstrak poin-poin utama dan memberikan ringkasan singkat untuk artikel yang panjang.

Instruksi Ringkaslah teks ini dalam tiga kalimat.

Input Daylight Savings Time (DST) - proses memajukan jam
satu jam di musim panas - dimulai di Jerman pada tahun 1916 ...
Keluaran _____

Tugas ini mengharuskan LLM untuk memahami instruksi "Ringkas teks ini dalam tiga kalimat" dan melakukan ringkasan yang sesuai. Namun, LLM biasanya dilatih untuk token berikutnya prediksi daripada menghasilkan keluaran yang mengikuti instruksi. Menerapkan LLM yang telah dilatih sebelumnya contoh di atas kemungkinan besar akan menyebabkan model terus menulis artikel input sebagai gantinya merangkum poin-poin utama. Tujuan dari penyelarasan instruksi (atau penyempurnaan instruksi) adalah untuk menyetel LLM agar merespons instruksi dan maksud pengguna secara akurat. Sisa bagian ini akan membahas beberapa isu yang berkaitan dengan penyelarasan instruksi, termasuk penyempurnaan LLM untuk mengikuti instruksi, menghasilkan atau mengumpulkan data instruksi, dan menggeneralisasi penyelarasan instruksi.

4.2.1 Penyetelan Halus yang Diawasi

Salah satu pendekatan langsung untuk mengadaptasi LLM agar sesuai dengan instruksi adalah dengan menyempurnakan model-model ini menggunakan pasangan input-output beranotasi [Ouyang et al., 2022; Wei et al., 2022a]. Tidak seperti pelatihan model bahasa standar, di sini kami tidak ingin memaksimalkan kemungkinan menghasilkan model bahasa yang lengkap. urutan, tetapi lebih pada memaksimalkan kemungkinan menghasilkan sisa urutan yang diberikan awalan. Pendekatan ini membuat penyempurnaan instruksi sedikit berbeda dari pra-pelatihan. SFT data adalah kumpulan pasangan input-output (dilambangkan dengan S), dimana setiap output adalah benar respons untuk instruksi input yang sesuai. Misalnya, di bawah ini adalah kumpulan data SFT

x (instruksi + masukan pengguna)	y (keluaran)
Rangkum artikel berikut. Artikel: Dalam beberapa tahun terakhir, energi surya telah mengalami pertumbuhan yang belum pernah terjadi sebelumnya, menjadi yang paling cepat berkembang ...	{yringkasany}
Ekstrak angka-angka keuangan utama dari laporan pendapatan berikut. Laporan: Perusahaan melaporkan pendapatan sebesar \$10 juta pada kuartal pertama dengan margin keuntungan 15% ...	Pendapatan: \$10 juta, Margin Keuntungan: 15%
Klasifikasikan email berikut sebagai spam atau bukan spam. Teks: Selamat! Anda telah memenangkan kartu hadiah senilai \$500. Klik di sini untuk mengklaim sekarang.	Spam
Berikan solusi untuk masalah teknis berikut. Masalahnya: komputer saya berjalan lambat dan sering macet.	Pertama, periksa ...

di mana instruksi disorot. Kumpulan data ini berisi instruksi dan output terkait untuk beberapa masalah NLP yang berbeda, sehingga kita dapat menyempurnakan LLM untuk menangani beberapa tugas secara bersamaan.

Misalkan $x = x_0 \dots x_m$ adalah urutan input (misalnya, instruksi + input pengguna) dan $y = y_1 \dots y_n$ adalah urutan output yang sesuai. Dalam SFT, kami bertujuan untuk memaksimalkan probabilitas output y yang diberikan input x . Pertimbangkan LLM dengan parameter yang telah dilatih sebelumnya $\tilde{\theta}$. Tujuan fine-tuning kemudian dapat dirumuskan sebagai:

$$\tilde{\theta} = \arg \max_{\theta^+} \log \Pr_{\theta^+}(y|x) \quad (4.1)$$

di mana $\tilde{\theta}$ menunjukkan parameter yang dioptimalkan melalui penyetelan halus, dan θ^+ menunjukkan penyesuaian terhadap $\tilde{\theta}$. Di sini kita akan menghilangkan superskrip $+$ dan menggunakan θ untuk mewakili θ^+ agar notasi tetap rapi. Namun, pembaca harus mengingat bahwa penyempurnaan dimulai dari parameter yang telah dilatih sebelumnya, bukan parameter yang diinisialisasi secara acak.

Fungsi objektif log $\Pr_{\theta}(y_i | x, y < i)$ dihitung dengan menjumlahkan log-probabilitas token di y , bersyarat pada input x dan semua token sebelumnya $y < i$:

$$\log \Pr_{\theta}(y|x) = \sum_{i=1}^N \log \Pr_{\theta}(y_i | x, y < i) \quad (4.2)$$

Formulasi ini setara dengan meminimalkan kerugian entropi silang.

Perhatikan bahwa meminimalkan log-probabilitas bersyarat log $\Pr_{\theta}(y|x)$ bukanlah masalah pelatihan model bahasa standar. Jika kita menggabungkan x dan y sebagai satu deret tunggal, bentuk pemodelan bahasa yang lebih umum didasarkan pada log-probabilitas gabungan log $\Pr_{\theta}(x, y)$, yaitu, kita meminimalkan kerugian atas semua token deret $\text{seq}_{x,y} = [x, y]$. Kita dapat menulis probabilitas deret ini menggunakan aturan rantai

$$\log \Pr_{\theta}(x, y) = \log \Pr_{\theta}(x, y) = \log \Pr_{\theta}(x) + \sum_{i=1}^n \log \Pr_{\theta}(y_i | x, y_{<i}) \quad (4.3)$$

ditarik ke 0 perhitungan kerugian

Ada dua suku di sisi kanan persamaan. Kita dapat dengan mudah menetapkan suku pertama $\log \Pr_{\theta}(x)$ menjadi 0, dengan hanya berfokus pada suku kedua $\log \Pr_{\theta}(y|x)$ untuk perhitungan kerugian. Hasilnya, pelatihan dapat diimplementasikan menggunakan LLM standar. Untuk urutan $\text{seq}_{x,y}$, pertama-tama kita jalankan forward pass seperti biasa. Kemudian, selama backward pass, kita paksa kerugian yang terkait dengan x menjadi nol. Gambar 4.2 menunjukkan ilustrasi proses ini.

Dengan mengambil $\log \Pr_{\theta}(\text{seq}_{x,y})$ sebagai fungsi objektif, kita dapat mendeskripsikan SFT menggunakan bentuk pelatihan model bahasa biasa:

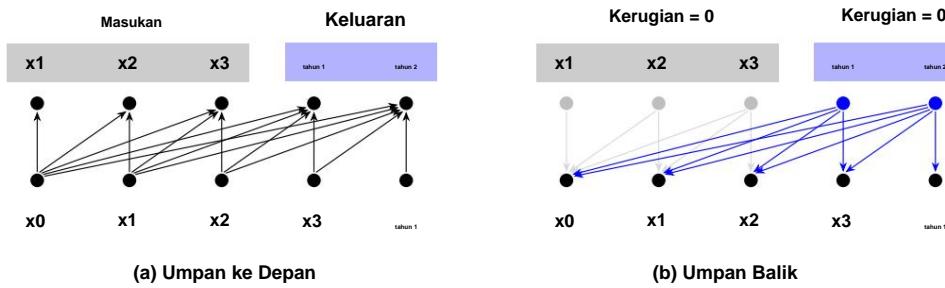
$$\tilde{\theta} = \arg \max_{\theta} \log \Pr_{\theta}(\text{seq}_{x,y}) \quad (4.4)$$

Masalah yang kami pertimbangkan di atas pada dasarnya adalah masalah prediksi satu putaran, di mana LLM menghasilkan respons berdasarkan satu masukan tanpa interaksi atau umpan balik lebih lanjut dari pengguna.

Masukan diproses, dan keluaran dihasilkan sekaligus. Hal ini biasa terjadi dalam

4.2 Penyelarasan Instruksi

159



Gambar 4.2: Ilustrasi fine-tuning terbimbing untuk LLM. Kami menggabungkan input dan output menjadi satu urutan. Selama forward pass, kami menjalankan LLM seperti biasa. Selama backward pass, kami menghitung kerugian hanya untuk bagian keluaran dan cukup atur kerugian pada bagian masukan ke 0.

skenario di mana satu pertanyaan diajukan, dan satu jawaban diberikan, tanpa tindak lanjut pertanyaan atau klarifikasi. Namun, dalam praktiknya, terkadang kita harus menangani masalah prediksi multi-putaran, misalnya, LLM terlibat dalam dialog selama beberapa putaran. Dalam pengaturan ini, LLM tidak hanya menghasilkan respons berdasarkan masukan awal tetapi juga menggabungkan masukan berikutnya masukan yang mungkin menyempurnakan atau memperluas interaksi sebelumnya. Misalnya, kita dapat menggunakan LLM untuk bertindak sebagai chatbot bantuan perawatan kesehatan dan melakukan percakapan dengan pengguna, seperti ini

Pengguna Saya merasa sangat lelah akhir-akhir ini.

Chatbot Saya turut prihatin mendengarnya. Selain merasa lelah, apakah Anda merasakan gejala lain?
gejala lainnya?

Pengguna Ya, saya juga sering mengalami sakit kepala.

Chatbot Sudah berapa lama gejala ini berlangsung?

Pengguna Sekitar seminggu sekarang.

Chatbot Mungkin ada baiknya untuk berkonsultasi dengan profesional kesehatan.

Anda suka bantuan membuat janji temu?

Pengguna Ya, silakan. Apakah bisa setelah jam kerja?

Chatbot Tentu, saya bisa mengturnya. Ada slot yang tersedia Rabu depan

dan Kamis setelah pukul 5 sore. Hari mana yang lebih cocok untuk Anda?

...

Dalam tugas ini, ada beberapa putaran percakapan, yang masing-masing melibatkan pembuatan respons berdasarkan permintaan atau pertanyaan pengguna dan riwayat percakapan. Misalkan kita memiliki K putaran percakapan, dilambangkan dengan $\{x^1, \dots, x^K, y^K\}$. Di sini x^k dan y^k menunjukkan permintaan pengguna dan respons, masing-masing, untuk setiap putaran k . Log-probabilitas menghasilkan Jawabannya dapat dituliskan sebagai $\log P(y^k | x^1, \dots, x^{k-1})$. Tujuan kita adalah memaksimalkan jumlah dari log probabilitas ini

$$\tilde{y} = \arg \max_{\tilde{y}} \text{logaritma } P(y^k | x^1, \dots, x^{k-1}) \quad (4.5)$$

Implementasi langsung dari hal ini melibatkan penghitungan probabilitas bersyarat untuk setiap k. Namun, hal ini memerlukan menjalankan LLM sebanyak K kali, setiap kali dengan riwayat percakapan yang lebih banyak untuk membuat prediksi. Metode yang lebih efisien adalah melakukan penghitungan kerugian dari semua respons dalam satu kali menjalankan LLM. Untuk melakukan ini, kami $\text{seq}x_1, y_1, \dots, x_K, y_K = [x_1, \dots, x_K]^T$ merepresentasikan percakapan sebagai urutan x , y K] (atau singkatnya seq). Log-probabilitas diberikan oleh

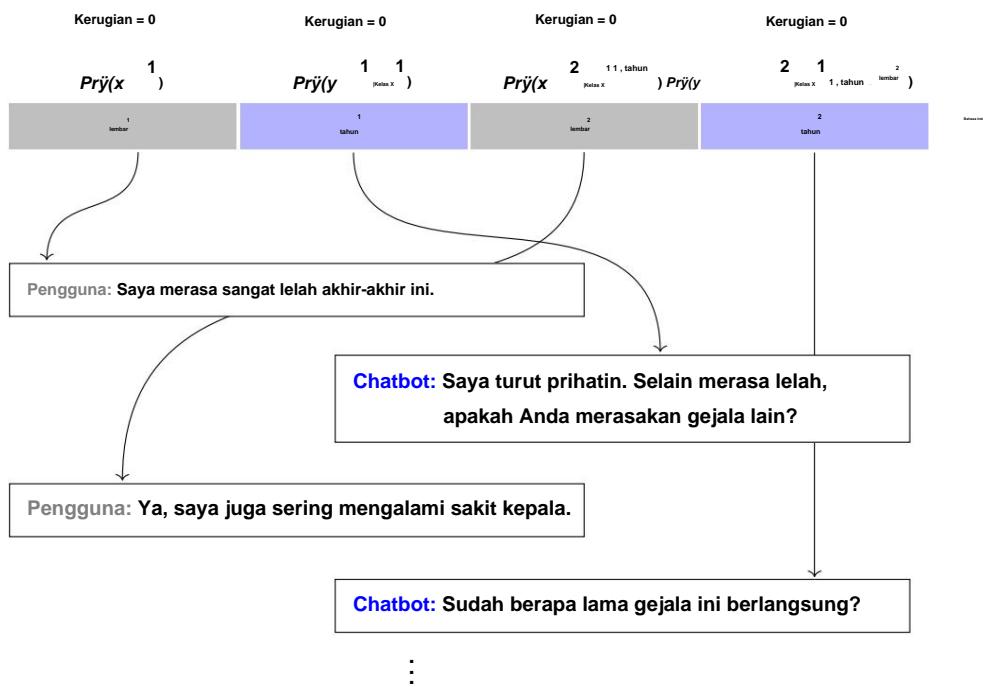
$$\begin{aligned}\log P_{\hat{y}}(\text{seq}) &= \log P_{\hat{y}}(x_1, \dots, x_K, y_K) \\ &= \log P_{\hat{y}}(x_1) + \log P_{\hat{y}}(y_1 | x_1) + \dots + \\ &\quad \underbrace{\log P_{\hat{y}}(x_K)}_{\text{diatur ke } 0} + \underbrace{\log P_{\hat{y}}(y_K | x_1, \dots, x_{K-1})}_{\text{perhitungan kerugian}} + \\ &\quad \underbrace{\log P_{\hat{y}}(x_K | x_1, \dots, x_{K-1})}_{\text{diatur ke } 0} + \\ &\quad \underbrace{\log P_{\hat{y}}(y_K | x_1, \dots, x_{K-1})}_{\text{perhitungan kerugian}} \quad (4.6)\end{aligned}$$

Triknya di sini adalah kita mengabaikan kerugian dalam menghasilkan masukan pengguna, seperti yang diilustrasikan dalam Gambar 4.3. Oleh karena itu, kami hanya menghitung probabilitas untuk menghasilkan respons berdasarkan riwayat percakapannya, dengan kata lain, nilai pada sisi kanan Persamaan (4.6) sebenarnya sama dengan nilai pada sisi kanan Persamaan (4.5). Seperti pada Persamaan (4.4), pelatihan model prediksi multi-putaran ini dapat dicapai dengan memaksimalkan log likelihood pada set data pelatihan D:

$$\hat{y} = \arg \max_{\hat{y}} \sum_{\text{seq} \in D} \log P_{\hat{y}}(\text{urutan}) \quad (4.7)$$

Meskipun penerapan metode SFT yang diperkenalkan di atas tampak sepele karena pada dasarnya sama dengan pelatihan model bahasa biasa, masih ada masalah yang perlu dipertimbangkan dalam praktiknya. Misalnya,

- SFT memerlukan data berlabel. Hal ini membuat SFT sangat berbeda dari pra-pelatihan, di mana teks mentah digunakan sebagai data pelatihan dan tersedia secara luas. Seperti dalam masalah pembelajaran mesin terbimbing lainnya, anotasi dan pemilihan data dalam SFT bukanlah tugas yang sederhana. Secara umum, kami ingin mengembangkan data SFT yang jumlahnya banyak dan berkualitas tinggi, dan data ini harus sangat relevan dengan tugas yang akan dilakukan LLM. Di sisi lain, ada kebutuhan untuk menyempurnakan LLM dengan lebih sedikit data untuk meminimalkan biaya komputasi dan konstruksi data. Sering kali, kualitas LLM sangat bergantung pada data yang digunakan dalam SFT. Dengan demikian, data tersebut harus dikembangkan dan diperiksa dengan cermat. Seperti yang akan kita lihat di subbagian selanjutnya, SFT dapat menjadi lebih efisien dan efektif.
- SFT masih membutuhkan komputasi yang mahal untuk LLM karena ukurannya yang besar. Akibatnya, pemeliharaan dan pembaruan model tersebut membutuhkan banyak sumber daya. Misalnya, penerapan pembaruan gradien pada miliaran parameter dalam LLM memerlukan daya komputasi dan memori yang signifikan. Hal ini sering kali memerlukan lingkungan komputasi berkinerja tinggi, yang mahal untuk dioperasikan. Untuk mengatasi tantangan ini, berbagai strategi pengoptimalan, seperti pemangkas, kuantisasi, dan penggunaan algoritme pelatihan yang lebih efisien, telah dieksplorasi. Secara khusus, ada minat yang signifikan terhadap metode fine-tuning yang hemat parameter yang dirancang untuk mempertahankan kinerja terkini tanpa memerlukan sumber daya komputasi yang besar. Kita telah melihat di Bab 3 bahwa penerapan teknik seperti soft prompt



Gbr. 4.3: Ilustrasi fine-tuning terbimbing untuk model percakapan. Di sini LLM bertindak sebagai chatbot untuk menanggapi setiap permintaan berdasarkan riwayat percakapan. Percakapan berlangsung dengan bergantian antara pengguna dan chatbot. Dalam SFT, kami memperlakukan seluruh percakapan sebagai urutan, seperti dalam LLM standar, tetapi menghitung kerugian hanya untuk tanggapan LLM.

membuat proses fine-tuning lebih efisien. Untuk pembahasan lebih lanjut tentang metode yang hemat parameter, pembaca dapat merujuk ke makalah terkait tentang masalah ini [Houlsby et al., 2019; Hu et al., 2022; Han et al., 2024].

- SFT dapat dianggap sebagai langkah pasca-pelatihan setelah pra-pelatihan. Ini adalah fase pelatihan terpisah yang dirancang untuk mempertahankan keuntungan dari pra-pelatihan awal sambil menggabungkan penyesuaian baru. Ini mungkin tampak paradoks karena memperbarui LLM yang telah dilatih sebelumnya dengan data lebih lanjut berpotensi menyebabkan model melupakan sebagian pengetahuan sebelumnya. Bayangkan sebuah skenario di mana kita memiliki sejumlah besar data SFT dan menyempurnakan LLM secara ekstensif. Dalam kasus ini, LLM dapat menyesuaikan data secara berlebihan, yang pada gilirannya dapat mengurangi kinerja generalisasi atau menyebabkan lupa yang parah. Strategi umum untuk mengurangi masalah ini adalah dengan menggunakan teknik regularisasi dan penghentian awal. Pendekatan praktis lainnya adalah dengan menggunakan laju pembelajaran yang lebih kecil untuk menyesuaikan bobot LLM dengan hati-hati. Selain itu, penyempurnaan dengan data dari berbagai sumber dan domain masalah juga dapat bermanfaat. Namun demikian, dalam praktiknya, langkah SFT sering kali diperiksa dengan cermat dan memerlukan upaya rekayasa dan eksperimental yang substansial untuk m-

4.2.2 Penyempurnaan Akuisisi Data

Penyempurnaan data sangat penting sehingga banyak pekerjaan terkini dalam LLM difokuskan pada pengembangan berbagai set data untuk penyempurnaan instruksi. Seperti kebanyakan pekerjaan dalam pembelajaran mesin, secara umum ada dua pendekatan untuk akuisisi data — pembuatan data manual dan pembuatan data otomatis.

4.2.2.1 Data yang Dihasilkan Secara Manual

Salah satu metode yang mudah adalah merekrut anotator manusia untuk membuat pasangan input-output untuk tugas yang diinginkan. Tidak seperti anotasi data dalam NLP konvensional, seperti klasifikasi teks, di mana anotator hanya menetapkan label pada teks yang dikumpulkan sesuai dengan pedoman, membuat data fine-tuning untuk LLM memerlukan lebih banyak langkah dan upaya, sehingga membuatnya lebih menantang. Misalkan kita ingin memperoleh data fine-tuning untuk tugas penerjemahan mesin dari Inggris ke Mandarin. Langkah pertama adalah menulis templat perintah untuk menjelaskan tugas dan memformat masalah dengan jelas.

Instruksi Terjemahkan teks dari bahasa Inggris ke bahasa Mandarin.

Masukan Pengguna {text}

Keluaran {translation}

Kemudian, kami mengumpulkan pasangan teks sumber dan target (yaitu, teks bahasa Mandarin dan terjemahannya yang sesuai), dan mengganti variabel {text} dan {translation} untuk menghasilkan sampel fine-tuning. Misalnya, diberikan sepasang kalimat bahasa Inggris dan bahasa Mandarin

Bagaimana cuaca hari ini? → yyyy/yyyy {text} {translation}

kita dapat menghasilkan sampel fine-tuning menggunakan template prompt, seperti ini

Instruksi Terjemahkan teks dari bahasa Inggris ke bahasa Mandarin.

Masukan Pengguna Bagaimana cuaca hari ini?

Keluaran ?

Yaitu,

x = Terjemahkan teks dari Bahasa Inggris ke Bahasa Mandarin.\n Bagaimana cuaca hari ini? y = yyyy/yyyy

Kita dapat menggunakan pasangan (x, y) ini untuk menyempurnakan LLM, seperti dijelaskan pada subbagian sebelumnya.

Salah satu kesulitan di sini adalah terdapat banyak sekali cara berbeda untuk menulis templat perintah untuk tugas yang sama, dan orang yang berbeda dapat membuat templat perintah dengan kualitas dan kompleksitas yang berbeda-beda. Terkadang, kita mungkin menulis templat perintah dengan instruksi yang terlalu rumit atau bertele-tele. Terkadang, kita bahkan mungkin tidak tahu persis apa tugas targetnya dan bagaimana cara mendeskripsikannya. Strategi yang banyak diadopsi adalah membuat templat perintah untuk tugas NLP yang ada, mengingat telah ada begitu banyak masalah dan tolok ukur NLP yang mapan [Bach et al., 2022; Wang et al., 2022b; Mishra et al., 2022]. Dalam kasus ini, anotator dapat diberikan deskripsi tugas asli dan banyak contoh. Kemudian, mereka dapat menggunakan cara mereka sendiri untuk mengungkapkan cara meminta LLM untuk melakukan tugas tersebut. Perhatikan bahwa, meskipun metode tersebut dapat memudahkan proses pembuatan dan penulisan perintah, kita tetap memerlukan kerangka kerja anotasi dan sistem crowdsourcing untuk mengelola pekerjaan dan melakukan kontrol kualitas. Misalnya, kita umumnya perlu

dan format terpadu untuk menulis templat perintah, terutama ketika banyak anotator berkontribusi pada tugas yang sama. Salah satu keuntungan dari mendorong perintah dari tugas NLP yang ada adalah, setelah templat perintah dikembangkan, mudah untuk membuat perintah menggunakan contoh yang diberi anotasi dalam tugas asli. Misalnya, dengan dataset dwibahasa untuk terjemahan Inggris-ke-Mandarin, kita dapat dengan mudah membuat sejumlah contoh penyempurnaan dengan mengisi slot pada templat di atas dengan pasangan kalimat dalam dataset ini.

Pendekatan lain adalah dengan langsung menggunakan data yang tersedia secara alami di internet. Contoh umum adalah dengan mengumpulkan pasangan pertanyaan dan jawaban dari situs web QA untuk menyempurnakan LLM untuk tugas QA domain terbuka [Joshi et al., 2017]. Banyak tolok ukur dalam QA dibangun dengan cara ini karena ada begitu banyak jenis pertanyaan sehingga mustahil untuk memikirkan semuanya oleh sekelompok kecil orang. Sebaliknya, menggunakan data dari situs web tersebut dapat memastikan bahwa data penyempurnaan LLM berada pada tingkat yang baik atau dapat diterima dalam hal kuantitas dan kualitas.

Selain menggunakan sumber daya yang ada, cara mudah lainnya untuk mengembangkan kumpulan data yang dapat disesuaikan adalah dengan mengumpulkan data dari banyak orang. Pendekatan sederhana adalah dengan mengizinkan pengguna memasukkan pertanyaan apa pun, setelah itu tanggapan diberikan secara manual atau dibuat secara otomatis oleh LLM, lalu diberi anotasi dan dikoreksi secara manual. Dengan demikian, perilaku pengguna yang sebenarnya dapat ditangkap dan akibatnya dapat mengumpulkan masukan dan keluaran untuk sejumlah besar masalah "baru" yang tidak dapat dicakup oleh tugas NLP tradisional.

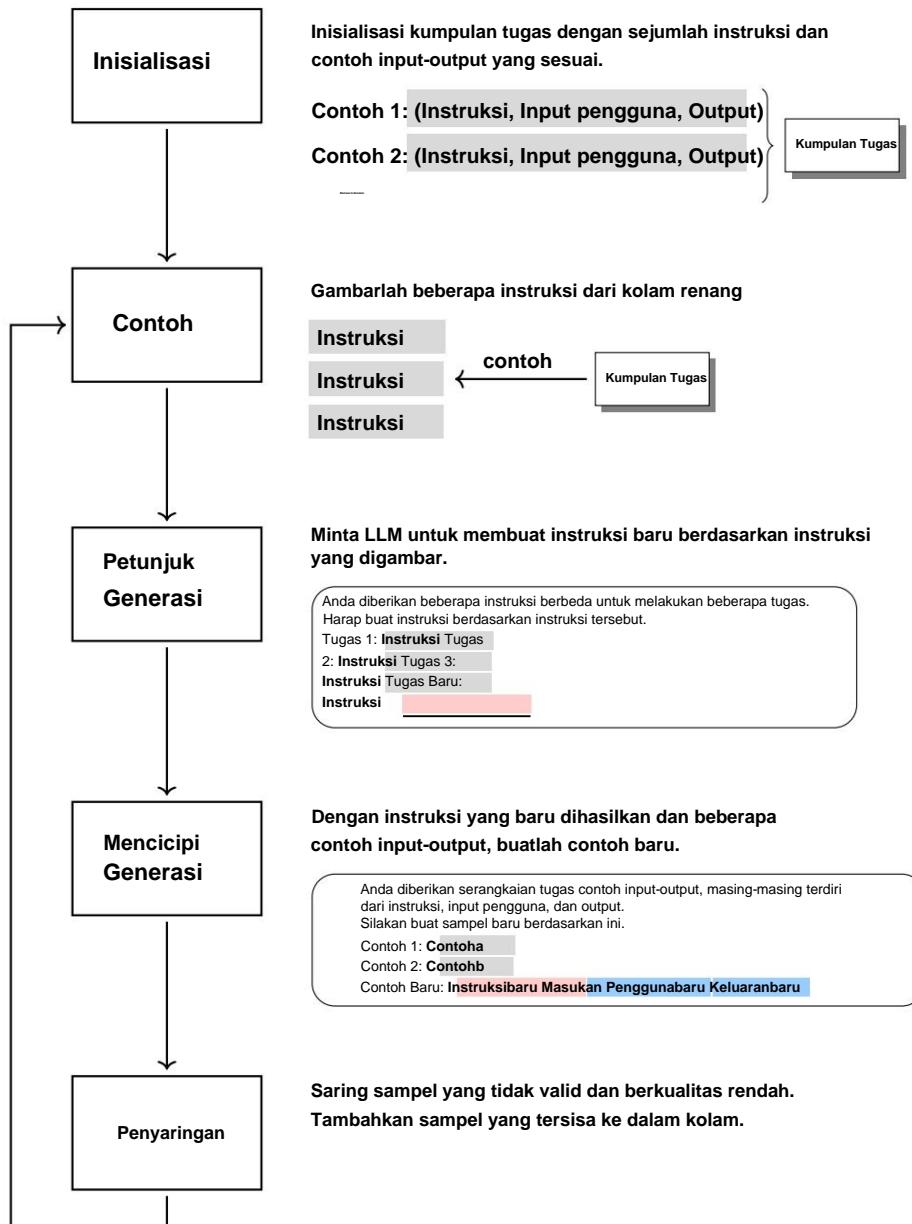
Masalah yang terkait dengan konstruksi kumpulan data fine-tuning adalah bahwa kita biasanya menginginkan data tersebut beragam se bisa mungkin. Banyak penelitian telah menemukan bahwa meningkatkan keragaman data fine-tuning dapat meningkatkan ketahanan dan kemampuan generalisasi LLM. Karena alasan ini, ada minat yang cukup besar untuk melibatkan perintah dan tugas yang lebih beragam dalam kumpulan data fine-tuning LLM. Kami akan memberikan pembahasan lebih lanjut tentang generalisasi fine-tuning di Bagian 4.2.4.

4.2.2.2 Data yang Dihasilkan Secara Otomatis

Salah satu keterbatasan pembuatan data manual adalah bahwa kualitas dan keragaman sangat bergantung pada pengalaman dan kreativitas manusia. Oleh karena itu, jika kita ingin LLM menangani berbagai tugas, yaitu, untuk secara efektif menjalankan instruksi apa pun, mengandalkan data yang dianotasi manusia untuk penyempurnaan LLM seringkali tidak efisien. Selain itu, cakupan data tersebut dapat dibatasi, dan data tersebut bahkan dapat mengandung bias yang diperkenalkan oleh para anotator itu sendiri. Pendekatan alternatif adalah dengan menghasilkan data secara otomatis. Misalnya, kita dapat mengumpulkan sejumlah pertanyaan melalui crowdsourcing, dan menggunakan LLM yang disetel dengan baik untuk menghasilkan jawaban atas pertanyaan tersebut. Pasangan pertanyaan-jawaban ini kemudian digunakan sebagai sampel penyempurnaan seperti biasa. Metode ini, meskipun sangat sederhana, telah diterapkan secara luas untuk menghasilkan data penyempurnaan skala besar untuk LLM.

Cara di atas untuk menghasilkan data fine-tuning sintetis serupa dengan yang digunakan dalam penambahan data untuk NLP. Jika kita memiliki LLM, kita dapat menghasilkan prediksi sebagai respons terhadap input apa pun. Mengulang proses ini untuk input yang berbeda memungkinkan kita membuat sampel fine-tuning dalam jumlah yang cukup. Metode seperti ini khususnya berguna untuk fine-tuning LLM baru menggunakan LLM yang telah disetel dengan baik. Namun, satu kelemahan dari pendekatan ini adalah pendekatan ini bergantung pada masukan yang dibuat atau dikumpulkan manusia untuk pembuatan data, yang mungkin ternyata tidak sesuai untuk menggeneralisasi LLM. Dalam banyak aplikasi LLM, tantangan signifikan muncul dari berbagai pertanyaan dan permintaan pengguna, yang banyak di antaranya tidak tercakup dalam tugas dan kumpulan data NLP yang ada. Dalam kasus ini, menjadi penting untuk menghasilkan tidak hanya prediksi tetapi juga masukan itu sendiri.

Di sini kami mempertimbangkan instruksi mandiri sebagai contoh untuk mengilustrasikan cara menghasilkan fine-tuning LLM



Gbr. 4.4: Ilustrasi instruksi mandiri [Wang dkk., 2023b]. Metode ini memelihara kumpulan instruksi dan sampel input-output yang sesuai. Awalnya, kumpulan tersebut berisi sejumlah instruksi dan sampel yang dibuat sendiri. Setiap kali, kami mengambil beberapa instruksi dari kumpulan tersebut. LLM kemudian diminta untuk membuat instruksi dan sampel baru berdasarkan instruksi dan sampel yang diambil. Akhirnya, instruksi dan sampel yang baru dibuat disaring dan ditambahkan ke kumpulan.

sampel [Wang et al., 2023d; Honovich et al., 2023]. Idenya adalah kita dapat mendorong LLM untuk membuat instruksi baru dengan belajar dari instruksi lain. Dengan instruksi ini, LLM kemudian dapat mengisi bidang lain (seperti masukan pengguna) dan menghasilkan prediksi. Gambar 4.4 menunjukkan ilustrasi skema instruksi mandiri. Di sini kami memberikan garis besar singkat dari langkah-langkah utama yang terlibat.

- Algoritme instruksi mandiri mengelola kumpulan tugas. Awalnya, algoritme ini berisi sejumlah tugas awal yang dibuat dengan tangan, masing-masing dengan instruksi dan contoh input-output. Saat algoritme berlanjut, instruksi dan contoh yang dihasilkan LLM akan ditambahkan ke kumpulan ini.

- Pada setiap langkah, sejumlah kecil instruksi diambil dari kumpulan instruksi. Misalnya, kita dapat secara acak memilih beberapa instruksi yang ditulis manusia dan beberapa instruksi yang dihasilkan LLM untuk memastikan keberagaman.
- Instruksi yang dipilih kemudian digunakan sebagai contoh demonstrasi. Dengan demikian, LLM dapat belajar dari contoh-contoh ini dalam konteksnya dan menghasilkan instruksi baru. Berikut ini adalah contoh templat untuk mendorong LLM.

Anda diberikan beberapa instruksi berbeda untuk melakukan beberapa tugas.

Silakan buat instruksi berdasarkan ini.

Tugas 1: {instruction1}

Tugas 2: {instruction2}

Tugas 3: {instruction3}

Tugas 4: {instruction4}

Tugas Baru: _____

- Berdasarkan instruksi yang dihasilkan, LLM kemudian diminta untuk melengkapi contoh dengan mengisi kolom input yang tersisa dan menghasilkan output yang sesuai. Berikut ini adalah contoh perintah.

Anda diberikan sekumpulan contoh input-output, yang masing-masing terdiri dari instruksi, input pengguna, dan output. Harap buat contoh baru berdasarkan contoh-contoh ini.

Contoh 1: {instruction1}

Masukan: {user-input1}

Keluaran: {output1}

Contoh 2: {instruction2}

Masukan: {user-input2}

Keluaran: {output2}

Contoh Baru: {new-instruction}

- Sampel yang baru dibuat ini diperiksa dengan beberapa aturan heuristik (seperti menyaring sampel atau instruksi yang mirip dengan yang sudah ada di kumpulan). Jika lolos, sampel dan instruksi ditambahkan ke kumpulan.

Proses pembuatan ini dapat diulang berkali-kali untuk memperoleh jumlah sampel fine-tuning yang memadai. Perhatikan bahwa, di atas, kami hanya menunjukkan templat perintah sederhana untuk membuat instruksi dan sampel fine-tuning. Tentu saja, kami dapat mengembangkan templat yang lebih baik untuk membuat instruksi dan sampel fine-tuning yang lebih beragam dan akurat. Misalnya, untuk tugas-tugas tertentu sepe

LLM mungkin cenderung menghasilkan prediksi yang bias, misalnya, sebagian besar sampel yang dihasilkan termasuk dalam satu kelas. Dalam kasus seperti itu, kita dapat menyesuaikan urutan pembuatan bidang yang berbeda. Lebih khusus lagi, kita dapat menentukan keluaran (yaitu, kelas) dengan beberapa prior, dan meminta LLM untuk menghasilkan masukan pengguna berdasarkan instruksi dan keluaran.

Metode ini menyerupai inversi masukan, di mana LLM menghasilkan masukan berdasarkan keluaran yang ditentukan [Longpre et al., 2023].

Penggunaan instruksi yang dihasilkan LLM dan sampel fine-tuning telah menjadi metode umum untuk mengembangkan LLM, terutama mengingat bahwa mengembangkan data tersebut secara manual sangat mahal sehingga sebagian besar kelompok penelitian tidak mampu membelinya. Dalam beberapa LLM yang disetel dengan baik, set data fine-tuning mereka mencakup sejumlah data sintetis, yang terbukti berguna [Ouyang et al., 2022; Taori et al., 2023; Chiang et al., 2023]. Telah ada penelitian lebih lanjut tentang pembuatan data sintetis untuk fine-tuning LLM. Misalnya, seseorang dapat menghasilkan instruksi yang lebih beragam dengan memperkenalkan algoritma evolusioner [Xu et al., 2024], atau menggunakan data sintetis sebagai sinyal supervisi dalam proses fine-tuning yang lebih maju [Chen et al., 2024b]. Baru-baru ini, ada juga minat yang cukup besar untuk menggunakan data sintetis dalam tahap pra-pelatihan [Gunasekar et al., 2023; Allal et al., 2023].

Dalam banyak aplikasi, skenario dunia nyata adalah, jika diberi tugas, kita dapat mengumpulkan atau membuat anotasi data fine-tuning dalam jumlah yang relatif kecil, misalnya, kita dapat merekrut pakar untuk membuat pertanyaan untuk tugas QA dalam domain tertentu. Namun, kuantitas dan keragaman data ini secara umum tidak mencukupi. Dalam kasus ini, kita dapat menggunakan teknik self-instruct untuk menghasilkan pasangan pertanyaan-jawaban yang lebih beragam, dan dengan demikian melengkapi data fine-tuning. Ini menyediakan cara untuk melakukan bootstrapping LLM mulai dari kumpulan benih sampel fine-tuning. Perhatikan bahwa menggunakan data yang dihasilkan sendiri adalah praktik umum dan telah lama diterapkan dalam NLP. Misalnya, pendekatan ini telah berhasil digunakan dalam parsing dan penerjemahan mesin [Charniak et al., 2000].

4.2.3 Penyetelan Halus dengan Data Lebih Sedikit

Dengan semakin menonjolnya fine-tuning instruksi, telah terjadi lonjakan permintaan untuk data fine-tuning skala besar dan berkualitas tinggi. Misalnya, set data fine-tuning FLAN, yang dikompilasi dari 1.836 tugas, berisi 15 juta sampel [Longpre et al., 2023]. Fine-tuning LLM dengan set data yang begitu besar biasanya merupakan tugas yang mahal secara komputasi, terutama mengingat bahwa memperbarui sejumlah besar parameter dalam LLM membutuhkan banyak sumber daya. Salah satu pendekatan untuk mengurangi masalah ini adalah dengan mengeksplorasi metode pelatihan model yang efisien, misalnya, seseorang dapat menggunakan metode yang hemat parameter untuk memperbarui hanya sebagian kecil dari model.

Namun, banyak set data fine-tuning berisi sejumlah besar data sintetis, di mana kesalahan dan bias masih tidak dapat dihindari.

Pendekatan lain untuk fine-tuning yang efisien adalah dengan hanya mempertimbangkan contoh yang paling relevan dan berdampak untuk fine-tuning. Dengan demikian, kita dapat mengurangi jumlah data yang perlu diproses sambil tetap mempertahankan kualitas pembaruan model. Ada beberapa metode untuk mencapainya. Misalnya, Zhou et al. [2023a] membangun dataset mengikuti instruksi yang hanya berisi 1.000 sampel dengan menyusun perintah dengan hati-hati dan mengumpulkan sampel dari berbagai tugas NLP. Mereka menunjukkan bahwa model LLaMa 65B yang disempurnakan dengan dataset ini dapat bersaing dengan atau bahkan lebih baik daripada model dengan upaya fine-tuning yang jauh lebih banyak. Ini menunjukkan bahwa LLM dapat diadaptasi untuk merespons berbagai tugas tanpa perlu fine-tuning pada semua jenis data mengikuti instruksi. Chen et al. [2024a] mengembangkan sistem berdasarkan model GPT-3.5 untuk menilai kualitas setiap sampel mengikuti instruksi. Oleh karena itu, mereka dapat memilih sampel berkualitas tinggi dari kumpulan data yang ada, yang menunjukkan kinerja fine-tuning yang lebih baik dengan sampel fine-tuning yang lebih sedikit. Para peneliti juga telah mengembangkan metode untuk memilih atau menyarankan

[Zhao et al., 2024; Ge et al., 2024], atau memprioritaskan data yang lebih signifikan memengaruhi proses fine-tuning [Xia et al., 2024]. Faktanya, sebagian besar metode ini dapat dilihat sebagai contoh dari keluarga besar metode pemilihan dan penyaringan data. Dan sering kali penggunaan data berkualitas lebih tinggi (tetapi mungkin lebih sedikit) bermanfaat untuk melatih model NLP.

Penemuan dalam fine-tuning instruksi agak berbeda dari pandangan tradisional dalam NLP: kemampuan model untuk menangani masalah kompleks dapat diaktifkan dengan sejumlah kecil data beranotasi, daripada memerlukan sejumlah besar data terbimbing untuk pelatihan ekstensif. Satu kemungkinan penjelasan adalah bahwa kemampuan menghasilkan respons yang benar berdasarkan instruksi telah dipelajari selama pra-pelatihan, tetapi pemetaan respons instruksi tersebut tidak dengan probabilitas tinggi selama inferensi. Fine-tuning dapat sedikit menyesuaikan model agar mengikuti instruksi, yang memerlukan upaya pelatihan yang jauh lebih sedikit daripada pra-pelatihan. Hal ini terkait erat dengan apa yang dikenal sebagai hipotesis penyelarasan superfisial, yang menyatakan bahwa pembelajaran terutama terjadi selama pra-pelatihan, dan fase fine-tuning atau penyelarasan berikutnya tidak berkontribusi secara signifikan pada basis pengetahuan dasar LLM [Zhou et al., 2023a]. Karena kemampuan inti dan pengetahuan model sudah terbentuk sejak pra-pelatihan, fine-tuning yang efektif untuk penyelarasan dengan kebutuhan pengguna dapat dicapai dengan upaya fine-tuning pelatihan yang relatif kecil. Hal ini menyiratkan kemungkinan penyempurnaan LLM dengan data yang sangat sedikit. Di sisi lain, mungkin tidak perlu membatasi penyempurnaan pada data respons-instruksi yang berpasangan. Misalnya, Hewitt et al. [2024] menemukan bahwa mengikuti instruksi dapat dicapai secara implisit dengan penyempurnaan LLM hanya pada respons, tanpa instruksi yang sesuai.

Konsep yang terkait dengan pembahasan di sini adalah efisiensi sampel. Metode pembelajaran mesin disebut efisien sampel jika dapat belajar secara efektif dari sejumlah kecil contoh pelatihan. Dalam pengertian ini, penyempurnaan instruksi efisien sampel dibandingkan dengan pra-pelatihan. Dari perspektif pembelajaran mesin, metode efisien sampel dapat dilihat sebagai cara yang efisien untuk mengambil sampel ruang data, dan menguntungkan karena memanfaatkan data langka secara optimal. Oleh karena itu, teknik pembelajaran berbasis pengambilan sampel, seperti banyak algoritma pembelajaran penguatan, dapat memperoleh manfaat dari pendekatan efisien sampel ini. Misalnya, dalam penyelarasan preferensi manusia, kita dapat mengambil sampel data preferensi secara efisien melalui model penghargaan [Liu et al., 2024b] atau meningkatkan efisiensi pengambilan sampel dalam pembelajaran kebijakan [Wang et al., 2024].

4.2.4 Generalisasi Instruksi

Dalam banyak masalah pembelajaran mesin dan NLP, melatih model untuk melakukan generalisasi merupakan tujuan mendasar. Misalnya, dalam klasifikasi teks, kami berharap model kami mengklasifikasikan teks baru yang tidak terlihat selama pelatihan dengan benar. Namun, generalisasi menimbulkan tantangan tambahan dalam penyempurnaan instruksi. Kami berharap LLM yang disempurnakan instruksi tidak hanya menghasilkan respons yang sesuai untuk berbagai masukan dalam suatu tugas, tetapi juga secara akurat melakukan berbagai tugas seperti yang dijelaskan oleh berbagai instruksi. Untuk mengilustrasikan masalah ini, pertimbangkan LLM $P(y|c, z)$, di mana c adalah instruksi, z adalah masukan pengguna, dan y adalah keluaran model yang sesuai (yaitu, respons). Misalkan kinerja model ini dievaluasi dalam bentuk metrik, yang ditulis sebagai $Kinerja(P(y|c, z))$ atau disingkat $P(c, z, y)$. Secara informal, ketika kami mengatakan model ini dapat melakukan generalisasi dalam \hat{y} , maksud kami mungkin ada nilai \hat{y} sehingga rata-rata tugas tertentu (ditunjukkan oleh instruksi c , kinerja pada masukan baru berada di atas nilai ini:

$$\frac{1}{n} \sum_{i=1}^n P(c, \hat{y}_i | z_i) > \bar{y} \quad (4.8)$$

di mana Z merupakan himpunan masukan baru, dan y adalah input dalam set ini dan yang sesuai dengan z keluaran, masing-masing.

Demikian pula, kita dapat mengatakan bahwa model ini dapat digeneralisasikan ke seluruh tugas jika kinerja rata-rata secara keseluruhan pasangan instruksi-input berada di atas beberapa \bar{y} :

$$\frac{1}{\text{Nilai c}} \sum_{\bar{y}, \bar{y} \text{, } z, y} P(k) > \bar{y} \quad (4.9)$$

di mana D adalah himpunan pasangan instruksi-masukan baru.

Di sini, kita perlu menangani variasi dalam dua dimensi: instruksi dan masukan pengguna. Hal ini membuat generalisasi menjadi sangat rumit, karena, secara intuitif, sebuah model perlu belajar dari sejumlah besar tugas dan berbagai pasangan masukan-keluaran yang terkait dengan setiap tugas untuk mencapai generalisasi yang baik. Seperti yang telah kita bahas beberapa kali dalam buku ini, mencapai generalisasi tersebut memerlukan biaya yang jauh lebih rendah daripada pra-pelatihan. Secara umum, penyempurnaan LLM dengan data respons instruksi sampai batas tertentu dapat menghasilkan model yang menghasilkan instruksi setelah tugas baru. Meskipun demikian, secara umum diyakini bahwa upaya tertentu masih diperlukan untuk mengadaptasi LLM agar dapat memahami dan menjalankan instruksi secara luas.

Bahasa Indonesia: Salah satu cara untuk menggeneralisasi fine-tuning instruksi adalah dengan meningkatkan keragaman data fine-tuning. Dalam studi sebelumnya tentang fine-tuning instruksi, peneliti mengembangkan banyak set data, yang mencakup berbagai macam tugas NLP dan instruksi yang berbeda untuk setiap tugas [Wang et al., 2022b; Sanh et al., 2022; Longpre et al., 2023]. Dengan mengubah tugas-tugas ini ke dalam format terpadu, seseorang dapat menyempurnakan LLM dengan jumlah sampel yang cukup besar, misalnya, ada beberapa set data fine-tuning instruksi yang melibatkan lebih dari 100 tugas NLP dan 1 juta sampel. Namun, set data awal ini sebagian besar berfokus pada masalah akademis yang ada, tetapi bukan yang ingin dihadapi pengguna dalam aplikasi dunia nyata. Banyak pekerjaan baru-baru ini telah mengalihkan fokus untuk mengatasi masalah baru dan lebih praktis. Misalnya, ada minat yang cukup besar dalam membangun set data yang berisi demonstrasi dan respons yang besar dan rumit dari model SOTA terhadap kueri pengguna nyata [Wang et al., 2023c; Teknium, 2023].

Mungkin penggunaan kumpulan data fine-tuning yang besar dan beragam berawal dari upaya untuk menskalakan LLM dalam dimensi yang berbeda. Memang, hukum penskalaan telah digunakan secara luas untuk memotivasi pengembangan berbagai macam LLM fine-tuning instruksi yang berbeda. Dan masuk akal untuk menskalakan fine-tuning instruksi agar LLM mengikuti instruksi yang luas. Namun, dari perspektif penyelarasan LLM, penskalaan fine-tuning instruksi mungkin tidak efisien untuk mencapai generalisasi.

Satu masalah adalah bahwa penyempurnaan instruksi bergantung pada pembelajaran terbimbing yang belajar untuk menggeneralisasi dan melakukan tugas berdasarkan pemetaan respons instruksi. Namun, pendekatan semacam itu tidak menangkap preferensi manusia yang halus atau kompleks (misalnya, nada, gaya, atau kualitas subjektif) karena ini sulit dikodekan sebagai data respons instruksi yang eksplisit. Selain itu, kinerja generalisasi dibatasi oleh keragaman dan kualitas kumpulan data respons instruksi. Mengingat keterbatasan ini, kami ingin menggunakan model preferensi sebagai langkah penyempurnaan tambahan setelah penyempurnaan instruksi, sehingga LLM dapat digeneralisasi lebih lanjut (lihat Bagian 4.3).

Pandangan lain adalah bahwa beberapa pemetaan respons instruksi mungkin sudah dipelajari selama pra-pelatihan, sehingga LLM yang telah dilatih sebelumnya telah mengodekan pemetaan tersebut. Namun, karena kita sering tidak tahu persis data apa yang digunakan dalam pra-pelatihan, sulit untuk menilai apakah kita perlu mempelajari pemetaan tersebut dalam penyempurnaan. Pertanyaan terkait adalah apakah generalisasi di luar distribusi

terutama dicapai selama pra-pelatihan atau penyempurnaan. Meskipun menjawab pertanyaan ini secara langsung berada di luar cakupan bab ini, telah ditunjukkan bahwa pra-pelatihan pada kumpulan data yang besar dan beragam efektif dalam meningkatkan kinerja di luar distribusi [Hendrycks et al., 2020; Radford et al., 2021; Gunasekar et al., 2023]. Hal ini menimbulkan masalah yang menarik: jika LLM telah dilatih dengan baik dalam skala besar, penyempurnaan mungkin tidak begitu penting untuk generalisasi di luar distribusi, karena model tersebut mungkin telah menemukan variasi distribusi yang cukup. Hal ini mendorong para peneliti untuk menyempurnakan LLM dengan upaya yang sederhana atau untuk mengeksplorasi metode baru untuk mencapai kepuasan yang dibahas di sub-bagian sebelumnya, misalnya, mengikuti instruksi dapat dihasilkan dengan menyempurnakan sejumlah kecil pasangan instruksi-respons yang dipilih secara cermat [Zhou et al., 2023a], atau bahkan dengan menggunakan metode yang tidak dirancang secara eksplisit untuk melakukannya [Kung dan Peng, 2023].

Pembahasan di atas memberikan dua strategi berbeda: satu memerlukan peningkatan skala dataset fine-tuning untuk keragaman yang lebih besar, yang lain memerlukan dataset fine-tuning yang kecil tetapi perlu untuk adaptasi LLM yang efisien. Namun, dalam praktiknya, melibatkan instruksi yang beragam sering kali membantu. Dalam banyak kasus, kita perlu mengadaptasi LLM kita untuk tujuan tertentu. Namun LLM, yang mungkin telah mengodekan pemetaan mengikuti instruksi yang luas selama pra-pelatihan, mungkin cenderung berperilaku sebagai pelaksana instruksi tujuan umum bahkan dengan fine-tuning yang sederhana. Fenomena yang menarik adalah bahwa ketika melakukan fine-tuning pada data matematika, LLM yang dihasilkan mungkin tidak mengkhususkan diri dalam keluaran matematika. Sebaliknya, model ini mungkin merespons instruksi umum secara normal, misalnya, dapat menghasilkan puisi jika diinstruksikan untuk melakukannya [Hewitt, 2024]. Ini bukan hal yang buruk, tetapi ini menunjukkan bahwa LLM mungkin tidak dengan mudah mengubah sifatnya dalam mengikuti instruksi umum. Dalam hal ini, adaptasi tambahan dengan data yang lebih beragam dapat membantu menyesuaikan cara LLM mengikuti instruksi, terutama untuk tugas-tugas yang ingin kita tangani.

4.2.5 Menggunakan Model Lemah untuk Memperbaiki Model Kuat

Sejauh ini, kami telah mengeksplorasi berbagai metode penyempurnaan instruksi berdasarkan data berlabel. Salah satu keterbatasan dari banyak metode tersebut adalah bahwa metode tersebut mengharuskan data diberi anotasi oleh manusia atau dihasilkan oleh LLM yang kuat, yang dapat memberikan sinyal supervisi yang akurat dalam penyempurnaan. Namun, dalam banyak kasus, LLM yang kami miliki sudah kuat (atau setidaknya menguntungkan dalam aspek-aspek tertentu dari pemecahan masalah), dan dengan demikian tidak mudah untuk menemukan model yang lebih unggul untuk supervisi. Bahkan bagi para ahli manusia, ketika masalahnya menjadi rumit, memberikan jawaban yang benar dan terperinci mungkin sulit, atau terkadang tidak layak. Misalnya, ketika dihadapkan dengan dokumen yang sangat panjang, para ahli akan merasa sulit untuk mengidentifikasi ketidakkonsistenan, bias halus, atau poin-poin penting yang hilang tanpa melakukan tinjauan.

Orang mungkin bertanya pada titik ini: dapatkah kita menggunakan LLM yang lemah untuk mengawasi LLM yang kuat? Ini tampaknya menjadi tantangan yang signifikan, tetapi ini mungkin mencerminkan skenario masa depan di mana kita perlu mengawasi sistem AI yang lebih cerdas daripada manusia atau sistem AI lainnya [Burns et al., 2023b]. Masalah penggunaan model yang lebih kecil dan kurang kompleks untuk meningkatkan pelatihan model yang lebih besar dan lebih kompleks juga disebut masalah generalisasi lemah-ke-kuat. Meskipun belum ada pendekatan yang matang untuk generalisasi lemah-ke-kuat, penggunaan model yang lebih kecil untuk membantu model yang lebih kuat memang terbukti berguna di beberapa area LLM.

Untuk penyempurnaan instruksi, salah satu cara paling sederhana untuk menerapkan LLM lemah adalah dengan menggunakan model ini untuk menghasilkan data penyempurnaan sintetis. Misalkan kita memiliki kumpulan input X , di mana setiap input mencakup instruksi dan input pengguna jika perlu. Untuk setiap $x \in X$, kita menggunakan LLM lemah $\text{Prw}(\cdot)$ untuk menghasilkan prediksi $y^* = \arg \max_y \text{Prw}(y|x)$. Kemudian, LLM kuat $\text{Prs}(\cdot)$ dapat

dilatih pada prediksi yang dihasilkan ini (lihat Persamaan (4.1)):

$$\tilde{y} = \arg \max_{\tilde{y}} \log \text{Pr}_{\theta}(\tilde{y}|x) \quad (4.10)$$

di mana \tilde{y} adalah parameter model.

Bentuk di atas mengubah masalah fine-tuning menjadi masalah penyulingan pengetahuan, dengan kata lain, kita menyinggung pengetahuan dari model yang lemah ke model yang kuat. Akibatnya, kita dapat menggunakan berbagai metode penyulingan pengetahuan untuk mencapai tujuan ini. Namun, menjelaskan fine-tuning dari lemah ke kuat dari perspektif penyulingan pengetahuan tidaklah mudah.

Kekhawatiran utama adalah bahwa model yang kuat mungkin hanya meniru atau melebih-lebihkan kesalahan model yang lemah dan gagal untuk digeneralisasi. Misalnya, model yang kuat yang telah disempurnakan tetap tidak dapat memecahkan masalah sulit yang tidak dapat diprediksi secara akurat oleh model yang lemah. Untungnya, eksperimen awal dalam bidang penelitian ini telah menunjukkan hasil yang positif dan menjanjikan. Misalnya, Burns dkk. [2023a] menemukan bahwa menyempurnakan model GPT-4 yang telah dilatih sebelumnya dengan supervisi tingkat GPT-2 dapat meningkatkan generalisasi di beberapa tugas NLP. Untuk mengukur bagaimana model yang lemah meningkatkan generalisasi model yang kuat, kami mendefinisikan istilah-istilah berikut:

- **Kinerja Lemah (Pweak).** Ini adalah kinerja set pengujian dari model yang lemah, yang dapat dianggap sebagai kinerja dasar.
- **Performa Lemah-ke-Kuat (Pweak \rightarrow kuat).** Ini adalah performa set uji dari yang kuat model yang disesuaikan dengan model yang lemah.
- **Performa Batas Atas yang Kuat (Pceiling).** Ini adalah performa set uji dari model kuat yang disempurnakan dengan data kebenaran dasar. Misalnya, kami menyempurnakan model kuat dengan prediksi yang diberi anotasi manusia dan mengambil model yang dihasilkan sebagai batas atas.

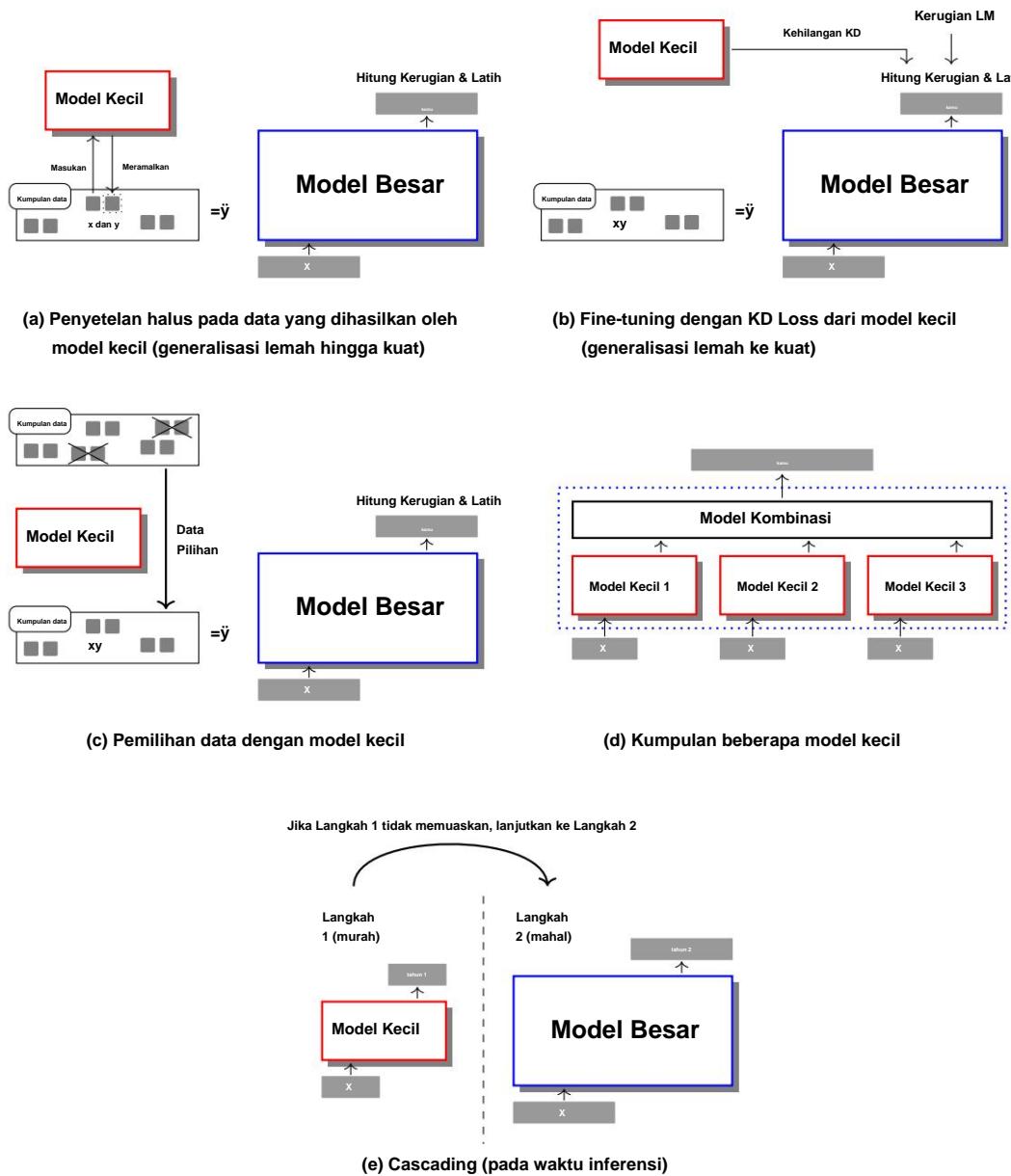
Kemudian, kesenjangan kinerja yang dipulihkan (PGR) dapat didefinisikan sebagai

$$\frac{\text{Pweak}\rightarrow\text{kuat} - \text{Pweak}}{\text{Pceiling} - \text{Pweak}} \quad (4.11)$$

Metrik ini mengukur seberapa besar kesenjangan kinerja antara model batas atas dan model lemah dapat dipulihkan oleh model lemah-ke-kuat. PGR sebesar 1 menunjukkan bahwa fine-tuning lemah-ke-kuat dapat sepenuhnya menutup kesenjangan kinerja, sedangkan PGR sebesar 0 menunjukkan tidak ada peningkatan. Dalam karya Burns et al. [2023a], ditunjukkan bahwa PGR dapat berada di sekitar 0,8 pada 22 tugas klasifikasi NLP. Perlu dicatat bahwa, meskipun potensi fine-tuning lemah-ke-kuat menjanjikan, mencapai generalisasi lemah-ke-kuat yang substansial tetap menjadi tujuan yang menantang yang memerlukan penyelidikan lebih lanjut [Aschenbrenner, 2024].

Penyetelan LLM dengan pengawasan lemah hanyalah salah satu pilihan untuk menggunakan model kecil guna meningkatkan model besar. Meskipun bagian ini terutama berfokus pada penyetelan LLM, kami juga menyebutkan metode lain di sini untuk memberikan pembahasan yang lebih lengkap (lihat Gambar 4.5 untuk ilustrasi metode ini).

- Daripada menggunakan model kecil untuk menghasilkan data sintetis, juga mudah untuk memasukkan kerugian penyulingan pengetahuan berdasarkan model ini. Misalnya, kerugian sederhana



Gbr. 4.5: Ilustrasi penggunaan model kecil untuk meningkatkan model besar dalam LLM. Salah satu pendekatan melibatkan penggunaan model yang lebih kecil untuk penyempurnaan atau pra-pelatihan model yang lebih besar. Ini termasuk menghasilkan data sintetis (a), menggabungkan kerugian tambahan (b), dan memilih data yang sesuai (c). Pendekatan lain melibatkan penggabungan model kecil dan model besar. Ini termasuk mempelajari model yang kuat dengan menggabungkan beberapa model kecil (d), dan menggabungkan model kecil dengan model besar (e).

fungsi yang mengukur perbedaan antara model kecil dan besar dapat didefinisikan sebagai:

$$\text{Rugi} = KL(Prw(\cdot|x) || \text{Prs } \tilde{y}(\cdot|x)) \quad (4.12)$$

Kemudian, kita dapat menambahkan kerugian ini ke kerugian asli pemodelan bahasa, dan menghasilkan tujuan pelatihan berikut

$$\tilde{y} = \arg \max_{\tilde{y}} \log \text{Prs } \tilde{y}(y|x) - \text{Kerugiankd}_{(x,y)\in D} \quad (4.13)$$

di mana D adalah himpunan pasangan masukan dan keluaran, dan \hat{y} adalah koefisien interpolasi. Metode ini dapat digunakan baik dalam fase pra-pelatihan maupun penyempurnaan. Kita dapat menyesuaikan \hat{y} untuk mengontrol seberapa besar model kecil memengaruhi pelatihan. Misalnya, kita dapat secara bertahap menurunkan \hat{y} agar pelatihan lebih bergantung pada kerugian pemodelan bahasa asli saat model besar menjadi lebih mampu.

- Pendekatan lain untuk melibatkan model kecil dalam pra-pelatihan dan penyempurnaan LLM adalah menggunakan untuk melakukan pemilihan atau penyaringan data. Dengan adanya suatu urutan, kita dapat menghitung kemungkinan atau entropi silang menggunakan model kecil. Kuantitas ini kemudian dapat digunakan sebagai kriteria untuk memilih atau menyaring data. Misalnya, urutan dengan kemungkinan rendah atau entropi silang tinggi mungkin dikecualikan dari set pelatihan, karena urutan tersebut kurang selaras dengan distribusi yang dipelajari model kecil. Sebaliknya, urutan dengan kemungkinan tinggi atau entropi silang rendah dapat diprioritaskan, memastikan bahwa pelatihan difokuskan pada data yang lebih relevan atau berkualitas tinggi.
- Pembelajaran ensemble adalah cara yang sederhana dan efektif untuk membangun model yang kuat dengan menggabungkan beberapa model yang lemah. Menerapkan teknik ini ke LLM sangatlah mudah. Kita dapat menggabungkan distribusi yang diprediksi oleh beberapa model kecil atau submodel khusus, dan memperoleh prediksi akhir dari hasil gabungan tersebut. Penggabungan ini dapat dilakukan dengan menggunakan metode seperti pemungutan suara mayoritas, rata-rata tertimbang, atau penumpukan.
- Model kecil juga dapat digunakan pada waktu inferensi untuk meningkatkan efisiensi secara keseluruhan. Misalkan kita memiliki model besar yang lambat tetapi lebih akurat, dan model kecil yang cepat tetapi kurang akurat. Dalam model cascading, model kecil terlebih dahulu memproses data input, dengan cepat menghasilkan hasil awal. Jika hasil ini memenuhi kriteria tertentu yang telah ditetapkan sebelumnya, hasil tersebut dapat langsung digunakan. Namun, jika hasil awal tidak cukup baik, input kemudian diteruskan ke model yang lebih besar dan lebih akurat untuk menghasilkan hasil yang lebih baik. Pendekatan ini secara signifikan mengurangi biaya komputasi dan latensi, karena model kecil dapat secara efektif menangani banyak input tanpa akses ke model besar.

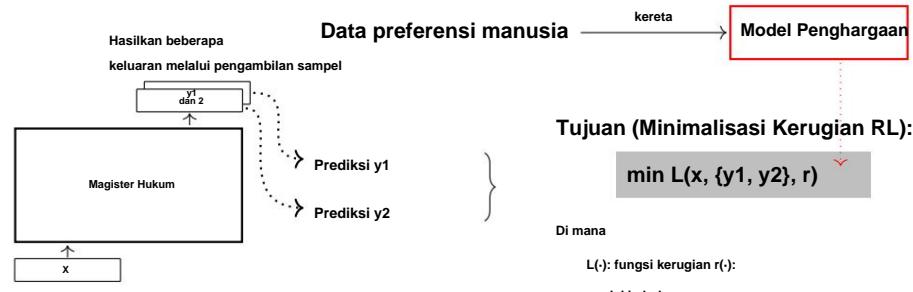
4.3 Penyelarasan Preferensi Manusia: RLHF

Sejauh ini dalam bab ini, kita telah berfokus pada penyempurnaan LLM menggunakan data berlabel berpasangan input-output. Pendekatan ini memungkinkan kita untuk mengadaptasi LLM untuk mengikuti instruksi melalui pembelajaran yang diawasi. Namun, dalam banyak aplikasi, LLM tidak hanya diminta untuk mengikuti instruksi tetapi juga untuk bertindak dengan cara yang lebih selaras dengan nilai dan preferensi manusia. Pertimbangkan skenario di mana seorang pengguna bertanya kepada LLM bagaimana cara meretas sistem komputer. Jika LLM tidak selaras dengan tepat, LLM dapat merespons dengan memberikan detail tentang cara melakukan aktivitas ilegal ini. Sebaliknya, respons yang lebih diinginkan mungkin adalah menasihati pengguna agar tidak terlibat dalam aktivitas ilegal dan menawarkan gambaran umum tentang konsekuensi dari tindakan tersebut. Kesulitan dalam mencapai hal ini adalah bahwa nuansa etika dan pertimbangan kontekstual yang diperlukan agar LLM dapat merespons dengan tepat dalam skenario seperti itu tidak selalu mudah untuk dikodekan ke dalam kumpulan data fine-tuning. Yang lebih menantang adalah, seringkali, manusia sendiri tidak dapat secara tepat mengekspresikan preferensi mereka sendiri.

Pada bagian ini, kami membahas metode fine-tuning LLM alternatif, yang disebut reinforcement learning from human feedback atau disingkat RLHF [Christiano et al., 2017; Stiennon et al., 2020]. Ide dasar di balik RLHF adalah bahwa LLM dapat belajar dari perbandingan keluaran model menggunakan reward



(a) Penyetelan halus yang diawasi (memaksimalkan probabilitas prediksi berdasarkan input)



(b) Pembelajaran Penguatan dari Umpaman Balik Manusia

Gbr. 4.6: Penyetelan halus terbimbing vs. pembelajaran penguatan dari umpan balik manusia. Dalam penyetelan halus terbimbing, kami mengoptimalkan LLM dengan memaksimalkan probabilitas prediksi berdasarkan masukan. Dalam pembelajaran penguatan dari umpan balik manusia, pertama-tama kami melatih model penghargaan pada data preferensi manusia (pada setiap pasangan prediksi, evaluator diminta untuk memilih mana yang mereka suka). Kemudian, kami menggunakan model penghargaan ini untuk mengawasi LLM selama penyelarasan.

model (lihat Gambar 4.6). Untuk melakukan ini, kita dapat merekrut pakar manusia yang menunjukkan preferensi mereka di antara pasangan keluaran yang dihasilkan oleh LLM. Data preferensi ini digunakan untuk melatih model penghargaan yang dapat memprediksi kualitas keluaran LLM yang dirasakan. Setelah dilatih, model penghargaan memberikan umpan balik dengan menetapkan skor pada keluaran baru yang dihasilkan LLM sebagai respons terhadap masukan. LLM menggunakan skor ini untuk memperbarui parameternya melalui algoritme pembelajaran penguatan. Di bagian selanjutnya dari bagian ini, pertama-tama kita akan memperkenalkan pengetahuan dasar tentang pembelajaran penguatan untuk memfasilitasi diskusi, dan kemudian membahas metode untuk melatih model penghargaan dan menyelaraskan LLM dengan model ini.

4.3.1 Dasar-dasar Pembelajaran Penguatan

Kita mulai dengan melihat beberapa konsep dasar pembelajaran penguatan. Perhatikan bahwa notasi yang digunakan di sini sedikit berbeda dari yang digunakan di bagian dan bab sebelumnya karena kami ingin membuat deskripsi kami lebih konsisten dengan yang ada dalam literatur pembelajaran penguatan. Meskipun demikian, kami akan menunjukkan bagaimana notasi ini sesuai dengan notasi pemodelan bahasa. Pembaca yang sudah familiar dengan teknik pembelajaran penguatan dapat melewati atau membaca sekilas subbagian ini.

Kerangka pembelajaran penguatan umum menjelaskan bagaimana agen berinteraksi dengan lingkungan yang dinamis. Interaksi ini dimodelkan sebagai serangkaian tindakan yang diambil oleh agen sebagai respons terhadap keadaan lingkungan. Pada setiap langkah waktu, agen mengamati keadaan saat ini, memilih tindakan berdasarkan kebijakannya, melakukan tindakan, dan kemudian menerima umpan balik dari lingkungan dalam bentuk hadiah dan keadaan baru. Urutan umpan balik mengamati-bertindak-menerima ini diulang

sampai agen mencapai tujuannya.

Sistem pembelajaran penguatan melibatkan beberapa komponen

- **Agen.** Ini adalah pembelajar atau pembuat keputusan dalam pembelajaran penguatan. Dalam konteks LLM, dapat dilihat sebagai LLM itu sendiri.
- **Lingkungan.** Ini mencakup segala sesuatu yang berada di luar agen yang berinteraksi dengan agen tersebut. Namun, lingkungan dalam LLM tidak terlalu berkaitan dengan ruang fisik atau virtual, melainkan lebih kepada kerangka kerja di mana agen (misalnya, LLM) menerima umpan balik dan belajar.
- **State (s).** State mewakili situasi lingkungan saat ini. Dengan urutan token untuk pemodelan bahasa, state pada langkah waktu dapat dilihat sebagai token yang kita amati sejauh ini, yaitu, token konteks yang kita ambil untuk memprediksi token berikutnya. Misalnya, kita dapat mendefinisikan $(x, y < t)$ sebagai state saat memprediksi token berikutnya pada langkah waktu t .
- **Tindakan (a).** Tindakan mewakili kemungkinan keputusan yang dapat diambil agen. Kita dapat melihatnya sebagai kemungkinan token yang diprediksi dalam kosakata.
- **Reward (R).** Reward adalah umpan balik dari lingkungan yang mengevaluasi keberhasilan suatu tindakan. Misalnya, $r(s, a, s')$ menunjukkan reward yang diterima agen karena mengambil tindakan a pada keadaan s dan pindah ke keadaan s' berikutnya. Kita dapat menunjukkan Jika urutan keadaan-tindakan diberikan, , reward pada langkah waktu t sebagai $r_t = r(st, proses pengambilan pada st+1)$. Perhatikan juga bahwa jika keputusan bersifat deterministik, kita dapat menghilangkan $st+1$ karena dapat ditentukan oleh st dan pada . Dalam kasus seperti ini, kita dapat menggunakan $r(st, at)$ sebagai singkatan untuk $r(st, st+1)$.
- **Kebijakan (\bar{y}).** Untuk LLM, kebijakan didefinisikan sebagai distribusi probabilitas atas token yang diprediksi oleh LLM, mengingat token konteks sebelumnya. Secara formal, ini dapat dinyatakan sebagai

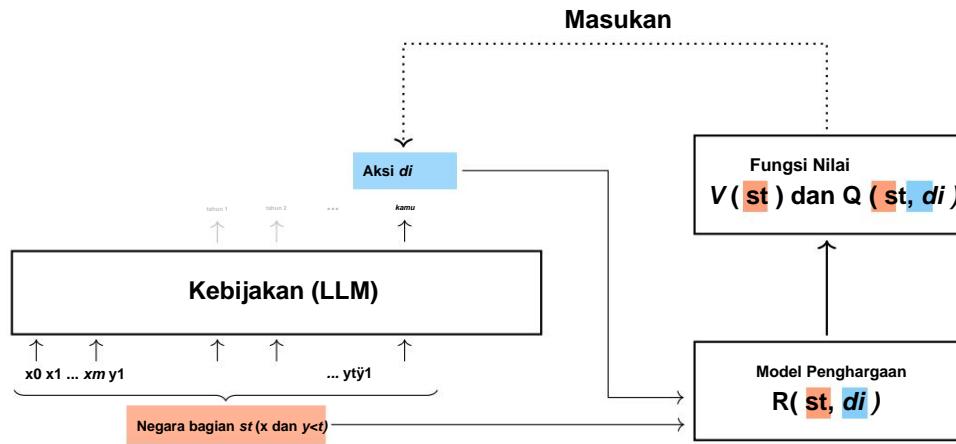
$$\bar{y}(a|s) = \Pr(yt | x, y < t) \quad (4.14)$$

di mana a berhubungan dengan token y , dan s berhubungan dengan konteks $(x, y < t)$. Gambar 4.7 mengilustrasikan bagaimana LLM dapat diperlakukan sebagai kebijakan dalam kerangka pembelajaran penguatan.

- **Fungsi Nilai (V dan Q).** Fungsi nilai-keadaan (atau fungsi nilai, singkatnya) menilai pengembalian yang didiskontokan (yaitu, imbalan yang terakumulasi) bagi agen yang dimulai dari keadaan tertentu s dan mengikuti kebijakan tertentu \bar{y} . Fungsi ini didefinisikan sebagai:

$$\begin{aligned} V(s) &= E(s_0, a_0, s_1) + \bar{y}(s_1, a_1, s_2) + \bar{y}^2(s_2, a_2, s_3) + \dots s_0 = s, \bar{y} \\ &= E = r_0 + \bar{y} = r_1 + \bar{y}^2 r_2 + \dots s_0 = s, \bar{y} \\ &= E \quad \text{jika } t \bar{y} rt s_0 = s, \bar{y} t=0 \end{aligned} \quad (4.15)$$

di mana $\bar{y} \in [0, 1]$ adalah faktor diskonto yang menyesuaikan pentingnya imbalan di masa depan, $s_0 = s$ menunjukkan bahwa agen memulai dengan keadaan s , dan ekspektasi E dilakukan pada semua lintasan yang mungkin (yaitu, urutan keadaan-tindakan). Demikian pula, fungsi nilai tindakan (atau



Gbr. 4.7: LLM sebagai kebijakan dalam pembelajaran penguatan. Pada setiap langkah t , LLM memprediksi token y_t berdasarkan masukan model x dan token $y_{<t}$ yang dihasilkan sebelumnya. Proses ini dapat dibingkai sebagai masalah pembelajaran penguatan, di mana y_t berfungsi sebagai tindakan, $(x, y_{<t})$ sebagai keadaan, dan distribusi yang diprediksi $Pr(y_t|x, y_{<t})$ sebagai kebijakan. Setelah y_t diprediksi, LLM memasukkan $(x, y_{<t})$ dan y_t ke model hadiah, yang mengevaluasi seberapa efektif token yang dipilih berkontribusi untuk mencapai hasil tekstual yang diinginkan. Evaluasi ini menghasilkan skor hadiah yang digunakan untuk menghitung fungsi nilai $V(st)$ dan $Q(st, at)$. Fungsi-fungsi ini kemudian memberikan umpan balik ke LLM dan memandu pelatihan kebijakan.

Fungsi nilai Q mengukur pengembalian yang diharapkan dimulai dari keadaan *mengambil* tindakan a dan kemudian mengikuti kebijakan \hat{y} , diberikan oleh

$$Q(s, a) = E = \text{bilangan bulat} \quad \text{jika } t \neq rt \text{ dan } s = s, a = a, \bar{y} \quad (4.16)$$

tanggal = 0

di mana $a_0 = a$ menunjukkan bahwa tindakan yang diambil pada keadaan awal adalah a .

Tujuan dari pembelajaran penguatan adalah untuk mempelajari kebijakan yang memaksimalkan imbalan kumulatif (atau pengembalian) yang diterima agen dalam jangka panjang. Diberikan urutan tindakan-kedua y = $\{(s_1, a_1), \dots, (s_T, a_T)\}$. Jumlah hadiah kumulatif atas urutan ini dapat ditulis sebagai

$$R(\ddot{y}) = \text{kanan} \quad (4.17)$$

Harapan dari imbalan kumulatif ini selama rentang urutan tindakan-keadaan diberikan dalam bentuk

$$\begin{aligned}
 J(\hat{y}) &= E_{\hat{y}\hat{y}D} R(\hat{y}) \text{ dan } \hat{y}\hat{y} \\
 &= \text{Persamaan kuadrat } Pr\hat{y}(\hat{y})R(\hat{y}) \\
 &\quad \hat{y}\hat{y}D \\
 &= T \\
 &= Pr\hat{y}(\hat{y}) \quad \text{kanan} \\
 &\quad \hat{v}\hat{v}D \quad \text{tanggall = 1}
 \end{aligned} \tag{4.18}$$

1Kami berasumsi urutan keadaan-tindakan dimulai dengan s_1 dan a_1 , bukan s_0 dan a_0 , agar sesuai dengan notasi yang umum digunakan dalam bab ini, di mana prediksi y biasanya dimulai dari y_1 . Tentu saja, urutan keadaan-tindakan juga umum dilambangkan sebagai $\{(s_0, a_0), \dots, (sT, aT)\}$ atau $\{(s_0, a_0), \dots, (sT\bar{y}_1, aT\bar{y}_1)\}$ dalam literatur. Namun, variasi notasi ini tidak memengaruhi pembahasan model yang disajikan di sini.

di mana \hat{y} menunjukkan bahwa \hat{y} diambil dari ruang urutan keadaan-tindakan D, dan subskrip \hat{y} menunjukkan parameter kebijakan. $J(\hat{y})$ juga disebut fungsi kinerja.

Maka tujuan pelatihannya adalah memaksimalkan $J(\hat{y})$:

$$\hat{y} = \arg \max_{\hat{y}} J(\hat{y}) \quad (4.19)$$

Sekarang, kita memiliki pendekatan pembelajaran penguatan sederhana: 1) kita mengambil sampel sejumlah urutan tindakan-keadaan; kemudian, 2) kita mengevaluasi setiap urutan menggunakan fungsi kinerja; kemudian, 3) kita memperbarui model untuk memaksimalkan fungsi kinerja ini. Jika kita mengambil Persamaan (4.18) dan menggunakan penurunan gradien untuk mengoptimalkan kebijakan, pendekatan ini akan membentuk suatu bentuk metode gradien kebijakan [Williams, 1992].

Perhatikan bahwa dalam banyak masalah NLP, seperti penerjemahan mesin, imbalan biasanya jarang. Misalnya, hadiah hanya diterima di akhir kalimat lengkap. Ini berarti bahwa $r_t = 0$ untuk semua $t < T$, dan r_t bukan nol hanya ketika $t = T$. Idealnya, seseorang mungkin lebih suka umpan balik yang segera dan sering (padat), dan dengan demikian pelatihan kebijakan dapat lebih mudah dan lebih efisien. Sementara beberapa metode telah diusulkan untuk mengatasi hadiah yang jarang, seperti pembentukan hadiah, kami akan melanjutkan pembahasan kami dengan mengasumsikan pengaturan hadiah yang jarang, di mana hadiah tersedia hanya setelah menyelesaikan prediksi.

Model yang dijelaskan dalam Persamaan (4.17-4.19) menetapkan bentuk dasar pembelajaran penguatan, dan banyak varian serta perbaikan model ini telah dikembangkan. Sebelum menunjukkan model yang lebih canggih tersebut, mari kita luangkan waktu sejenak untuk menafsirkan fungsi objektif $J(\hat{y})$ dari perspektif gradien kebijakan. Dalam penurunan gradien, kita perlu menghitung gradien $J(\hat{y})$ terhadap \hat{y} :

$$\begin{aligned} \frac{\text{Bahasa Indonesia:}}{\hat{y} \hat{y} \hat{y} \hat{y}} &= \frac{\hat{y}}{\hat{y} \hat{y} D} \frac{Pr(\hat{y}) R(\hat{y})}{\hat{y} \hat{y}} \\ &= \frac{\hat{y} Pr(\hat{y})}{R(\hat{y}) \hat{y} \hat{y}} \text{ Rumus} \\ &= \frac{\hat{y} D}{\hat{y} \hat{y} D} \frac{Pr(\hat{y})}{\frac{\hat{y} Pr(\hat{y}) / \hat{y} \hat{y}}{R(\hat{y})}} \text{ Rumus } R(\hat{y}) \\ &= \frac{Pr(\hat{y})}{\hat{y} \hat{y} D} \frac{\frac{\hat{y} Pr(\hat{y}) / \hat{y} \hat{y}}{R(\hat{y})}}{R(\hat{y})} \text{ Persamaan } R(\hat{y}) \\ &= \frac{Pr(\hat{y})}{\hat{y} \hat{y} D} \frac{\text{kuadrat } Pr(\hat{y}) \hat{y} \log Pr(\hat{y})}{R(\hat{y}) \hat{y} \hat{y}} \text{ Rumus} \end{aligned} \quad (4.20)$$

Dalam beberapa kasus, kita akan berasumsi bahwa setiap deret di D memiliki kemungkinan yang sama (yaitu, $Pr(\hat{y}) = \hat{y} \log Pr(\hat{y}) / |D|$). Dalam kasus ini kita dapat menyederhanakan Persamaan (4.20) dan hanya perlu mempertimbangkan suku dan $\hat{y} \hat{y} R(\hat{y})$:

$$\frac{\text{Bahasa Indonesia:}}{\hat{y} \hat{y} \hat{y} \hat{y}} = \frac{1}{m} \frac{\text{logaritma } Pr(\hat{y})}{R(\hat{y}) \hat{y} \hat{y}} \text{ Rumus} \quad (4.21)$$

Salah satu keuntungan dari hasil ini adalah bahwa $R(\hat{y})$ tidak perlu dapat dibedakan, yang berarti bahwa kita dapat menggunakan jenis fungsi penghargaan apa pun dalam pembelajaran penguatan.

Dengan memperlakukan pembangkitan urutan \hat{y} sebagai proses keputusan Markov, kita dapat lebih jauh

diturunkan $\frac{\hat{y} \log \Pr(\hat{y})}{\hat{y}\hat{y}}$ dan memperoleh:

$$\begin{aligned} \frac{\log \Pr(\hat{y})}{\hat{y}\hat{y}} &= \frac{T}{\text{logaritma } \hat{y}\hat{y} \text{ pada } /st \text{ Pr(st+1|st, pada)}} \\ &= \frac{\hat{y}}{\hat{y}\hat{y}} \frac{T}{\text{kebijakan}} \frac{\log \hat{y}\hat{y} \text{ pada } /st}{\hat{y}\hat{y}} \frac{T}{\text{dinamika}} \frac{\log \Pr(st+1|st, di)}{\hat{y}\hat{y}} \end{aligned} \quad (4.22)$$

di mana gradien dibagi menjadi dua bagian: gradien kebijakan dan gradien dinamika.

Komponen kebijakan, $\log \hat{y}\hat{y}(\text{at } /st)$, menentukan log-probabilitas mengambil tindakan *pada* saat tertentu dan

diparameterisasi oleh \hat{y} . Komponen dinamika, $\log \Pr(st+1|st, at)$, merepresentasikan status *st*, log-probabilitas transisi ke status *st+1* dari status *st* setelah mengambil tindakan *pada*. Dalam

pengaturan pembelajaran penguatan yang umum, dinamika tidak secara langsung dipengaruhi oleh parameter kebijakan \hat{y} , dan dengan demikian, turunannya sering kali nol. Oleh karena itu, dalam kasus ini, Persamaan (4.22) dapat disederhanakan menjadi

$$\frac{\log \Pr(\hat{y})}{\hat{y}\hat{y}} = \frac{\hat{y}}{\hat{y}\hat{y}} \frac{T}{\text{tanggall = 1}} \log \hat{y}\hat{y}(\text{pada } /st) \quad (4.23)$$

Dengan kata lain, kita hanya berkonsentrasi pada pengoptimalan kebijakan tanpa mempedulikan dinamika yang mendasarinya.

Dengan mensubstitusikan Persamaan (4.23) ke Persamaan (4.21), dan memperluas $R(\hat{y})$, kita memperoleh

$$\frac{\text{Bahasa Indonesia:}}{\hat{y}\hat{y}\hat{y}\hat{y}} = \frac{1}{|D|} \frac{\hat{y}}{\text{dari } \hat{y}\hat{y}D} \frac{T}{\text{tanggall = 1}} \frac{\log \hat{y}\hat{y}(\text{pada } /st)}{\text{kanan}} \frac{T}{\text{tanggall = 1}} \quad (4.24)$$

Meskipun pendekatan gradien kebijakan ini mudah, pendekatan ini memiliki masalah bahwa varians gradien yang diestimasikan bisa sangat tinggi, sehingga membuat proses pembelajaran menjadi tidak efisien dan tidak lancar. Salah satu alasan untuk masalah varians yang tinggi ini adalah bahwa imbalan dapat sangat bervariasi di berbagai langkah atau skenario. Bayangkan bahwa dalam rangkaian keputusan tindakan, model imbalan cenderung memberikan imbalan kecil untuk tindakan yang baik (misalnya, $Rt = 2$) dan hukuman besar untuk tindakan yang buruk (misalnya, $Rt = -10$). Skala imbalan yang bervariasi untuk tindakan baik dan buruk dapat menghasilkan imbalan total yang sangat rendah untuk keseluruhan rangkaian tindakan, bahkan jika rangkaian tersebut mencakup tindakan baik.

Satu metode sederhana untuk mengurangi varians gradien adalah dengan menetapkan garis dasar b dan mengurangkannya dari $t=1 rt$, yang ² menghilangkan bias sistematis dalam sinyal imbalan, yang membuat pembaruan lebih stabil dan kurang sensitif terhadap fluktuasi ekstrem dalam imbalan individual.

²Faktanya, penggunaan baseline b tidak mengubah varians total reward. Perlu dicatat bahwa meskipun memperkenalkan garis dasar tidak mengubah keseluruhan varians imbalan, hal itu membantu mengurangi varians estimasi gradien. Hal ini karena mengurangkan garis dasar dari total imbalan secara efektif mengurangi fluktuasi di sekitar nilai rata-ratanya, yang membuat estimasi gradien lebih stabil. Secara umum, operasi $rt - b$ memusatkan imbalan di sekitar nol (misalnya, $b = \frac{1}{T} \sum_{t=1}^T rt$), yang dapat menyebabkan berkurangnya varians dalam nilai y_t yang didefinisikan dari $\log \hat{y}\hat{y}(\text{at } /st)$ produk T .

Model gradien kebijakan dengan garis dasar ini dapat diberikan oleh

$$\begin{aligned}
 \frac{\text{Bahasa Indonesia:}}{\hat{y}J(\hat{y})\hat{y}} &= \frac{1}{|\mathcal{D}|} \sum_{\substack{\text{dari } \hat{y}y \\ \text{tanggall = 1}}}^T \log \hat{y}y(\text{pada } /st) \quad rt - b \\
 &= \frac{1}{|\mathcal{D}|} \sum_{\substack{\text{dari } \hat{y}y \\ \text{tanggall = 1}}}^T \log \hat{y}y(\text{pada } /st) \quad rk - b \\
 &= \frac{1}{|\mathcal{D}|} \sum_{\substack{\text{dari } \hat{y}y \\ \text{tanggall = 1}}}^T \log \hat{y}y(\text{pada } /st) \quad rk + \sum_{k=1}^{T-1} rk - b \quad (4.25)
 \end{aligned}$$

Di sini kita menulis $\sum_{k=1}^{T-1} rk$ sebagai jumlah dari dua suku $k=t$ dan $k=T$ yang membedakan antara imbalan yang diperoleh sebelum dan sesudah tindakan pada langkah waktu t . Perhatikan bahwa dalam proses pengambilan keputusan Markov, masa depan tidak bergantung pada masa lalu mengingat masa kini. Oleh karena itu, tindakan yang diambil pada langkah waktu t tidak dapat memengaruhi imbalan yang diterima sebelum t , atau dengan kata lain, imbalan sebelum t sudah "tetap" pada saat tindakan pada t dipilih. Suku $k=1 rk$ tidak berkontribusi pada gradien dan dapat dihilangkan, yang mengarah ke versi Persamaan (4.25) yang disederhanakan.

$$\frac{\text{Bahasa Indonesia:}}{\hat{y}J(\hat{y})\hat{y}} = \frac{1}{|\mathcal{D}|} \sum_{\substack{\text{dari } \hat{y}y \\ \text{tanggall = 1}}}^T \log \hat{y}y(\text{pada } /st) \quad rk - b \quad (4.26)$$

Perhatikan juga bahwa menghapus $\sum_{k=t}^{T-1} rk$ selanjutnya dapat mengurangi varians gradien.

Ada banyak cara untuk mendefinisikan baseline b . Di sini kita mempertimbangkan fungsi nilai dari state st , yaitu, nilai estimasi dari state st : $V(st) = E(rt + rt+1 + \dots + rT)$. Oleh karena itu kita memiliki

$$\begin{aligned}
 A(st, at) &= \sum_{k=t}^T rk - b \\
 &= rk - V(st) \quad (4.27)
 \end{aligned}$$

yang $k=t rk$ melambangkan pengembalian aktual yang diterima, dan $V(st)$ melambangkan pengembalian diharapkan. Di mana $A(st, at)$ (atau disingkat A_t) disebut keuntungan pada langkah waktu t , yang mengukur manfaat relatif dari tindakan at dibandingkan dengan nilai yang diharapkan dari mengikuti kebijakan dari keadaan st dan seterusnya.

Dengan menggunakan fungsi keuntungan $A(st, at)$, gradien $J(\hat{y})$ dapat dituliskan dalam bentuk

$$\frac{\text{Bahasa Indonesia:}}{\hat{y}J(\hat{y})\hat{y}} = \frac{1}{|\mathcal{D}|} \sum_{\substack{\text{dari } \hat{y}y \\ \text{tanggall = 1}}}^T \log \hat{y}y(\text{pada } /st)A(st, pada) \quad (4.28)$$

Tujuan optimasi ini sesuai dengan metode aktor-kritikus yang unggul (A2C) dalam pembelajaran penguatan [Mnih et al., 2016]. Dalam metode ini, aktor bertujuan untuk mempelajari suatu kebijakan. Ia memperbarui parameter kebijakan menggunakan Persamaan (4.28) untuk membantu lebih fokus pada tindakan yang mungkin meningkatkan kinerja. Di sisi lain, kritikus memperbarui estimasi fungsi nilai, yang digunakan untuk menghitung fungsi keunggulan $A(st, at)$, sehingga berfungsi sebagai evaluator

kebijakan yang dipelajari oleh aktor.

Dalam metode A2C, $A(st, at)$ biasanya dinyatakan sebagai perbedaan fungsi nilai tindakan $Q(st, at)$ dan fungsi nilai keadaan $V(st)$.

$$A(st, pada) = Q(st, pada) - V(st) \quad (4.29)$$

Sekilas, model ini mungkin tampak sulit dikembangkan karena memerlukan dua submodel terpisah untuk menghitung $Q(st, at)$ dan $V(st)$. Untungnya, mengingat $Q(st, at)$ dapat didefinisikan sebagai return $rt + V(st+1)$, kita dapat menulis ulang Persamaan (4.29) sebagai

$$A(st, pada) = rt + V(st+1) - V(st) \quad (4.30)$$

atau sebagai alternatif, memperkenalkan faktor diskon γ untuk mendapatkan bentuk yang lebih umum

$$A(st, pada) = rt + \gamma V(st+1) - V(st) \quad (4.31)$$

$A(st, at) = rt + \gamma V(st+1) - V(st)$ juga disebut kesalahan perbedaan temporal (TD). Yang kita perlukan adalah melalih jaringan kritik untuk fungsi nilai $V(st)$, dan kemudian menggunakan untuk menghitung fungsi keuntungan³.

Sampai di titik ini, kita telah menghabiskan banyak waktu untuk membahas dasar-dasar pembelajaran penguatan, khususnya tentang cara memperoleh tujuan pengoptimalan untuk metode A2C. Akan tetapi, pembelajaran penguatan merupakan bidang yang luas, dan banyak detail teknis yang tidak dapat dibahas di sini. Pembaca yang tertarik dapat merujuk ke buku-buku pembelajaran penguatan untuk mendapatkan informasi lebih rinci [Sutton dan Barto, 2018; Szepesvári, 2010]. Meskipun demikian, kita sekarang memiliki pengetahuan yang diperlukan untuk Pada subbagian berikutnya, kita akan kembali membahas penyelarasan LLM, dan menunjukkan cara menggunakan metode A2C untuk menyelaraskan dengan preferensi manusia.

4.3.2 Model Penghargaan Pelatihan

Kami telah menunjukkan bahwa model penghargaan memainkan peran yang sangat penting dalam kerangka pembelajaran penguatan umum dan membentuk dasar untuk menghitung fungsi nilai. Sekarang kami mempertimbangkan masalah pelatihan model penghargaan ini.

Dalam RLHF, model hadiah adalah jaringan saraf yang memetakan sepasang token input dan output urutan ke skalar. Diberikan input x dan output y , hadiah dapat dinyatakan sebagai

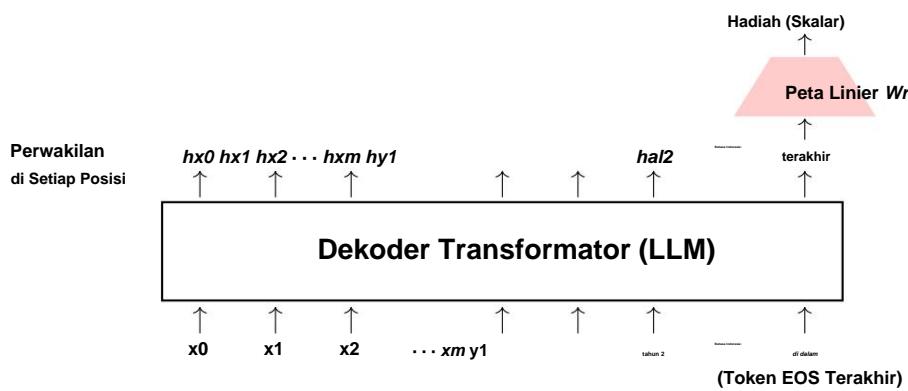
$$r = \text{Hadiah}(x, y) \quad (4.33)$$

di mana $\text{Reward}(\cdot)$ adalah model reward. r dapat diartikan sebagai ukuran seberapa baik output y selaras dengan perilaku yang diinginkan berdasarkan input x . Seperti yang dibahas di subbagian sebelumnya, baik x

³Kerugian pelatihan untuk jaringan nilai (atau jaringan kritik) dalam A2C secara umum diformulasikan sebagai kesalahan kuadrat rata-rata antara hasil yang dihitung $rt + \gamma V(st+1)$ dan nilai keadaan yang diprediksi $V(st)$. Misalkan jaringan nilai diparameterisasi oleh \hat{y} . Fungsi kerugian diberikan oleh

$$Lv(\hat{y}) = \frac{1}{M} \sum_{t=1}^T (rt + \gamma V(st+1)) - V(st) \quad (4.32)$$

di mana M adalah jumlah sampel pelatihan, misalnya, untuk urutan token T , kita dapat menetapkan $M = T$.



Gambar 4.8: Arsitektur model penghargaan berdasarkan Transformer. Komponen utama model ini masih berupa LLM. Kami menggunakan dekoder Transformer sebagai model representasi sekuens. Kami mengekstrak representasi posisi terakhir dari decoder sebagai representasi dari seluruh urutan $[x, y]$. Kami kemudian memetakan representasi ini ke skalar melalui transformasi linier, yang berfungsi sebagai skor hadiah untuk $[x, y]$.

dan y diasumsikan sebagai teks yang lengkap. Ini berarti bahwa model penghargaan mengevaluasi hubungan antara input dan output yang menyediakan konten semantik penuh. Misalnya, saat menerapkan model penghargaan, ia menetapkan nilai 0 (atau nilai lain yang telah ditentukan sebelumnya) di setiap posisi t dalam urutan keluaran $y = y_1 \dots y_n$. Hanya pada posisi akhir, ketika $t = n$, model penghargaan menghasilkan skor hadiah yang sebenarnya. Untuk menjaga notasi tetap rapi, kita akan menggunakan $r(x, y)$ untuk menunjukkan model hadiah $\text{Reward}(x, y)$ dari sini.

Ada banyak cara untuk menerapkan model penghargaan. Salah satu pendekatan sederhana adalah membangun model penghargaan berdasarkan LLM yang telah dilatih sebelumnya. Lebih khusus lagi, kita dapat menggabungkan x dan y untuk membentuk urutan token tunggal $\text{seq}, y = [x, y]$. Kami menjalankan LLM yang telah dilatih sebelumnya pada urutan ini, seperti biasa, dan pada setiap posisi, kita memperoleh representasi dari lapisan Transformer paling atas. Kemudian, kita ambil representasi pada posisi terakhir (dilambangkan dengan h_{last}) dan petakan ke skalar melalui linier transformasi:

$$r(x, y) = h_{\text{last}}W_r \quad (4.34)$$

di mana h_{last} adalah vektor berdimensi d , dan W_r adalah matriks pemetaan linier $d \times 1$. Arsitektur ini model penghargaan diilustrasikan pada Gambar 4.8.

Untuk melatih model penghargaan, langkah pertama adalah mengumpulkan umpan balik manusia pada serangkaian hadiah yang dihasilkan keluaran. Diberikan masukan x , kami menggunakan LLM untuk menghasilkan beberapa keluaran kandidat $\{y_1, \dots, y_N\}$. Umpan balik manusia dapat diperoleh dengan beberapa cara:

- Perbandingan Berpasangan (Pairwise Ranking). Diberikan dua output yang berbeda, pakar manusia pilih mana yang lebih baik.
- Rating. Pakar manusia memberikan skor atau rating untuk setiap output. Skor ini sering kali berupa nilai numerik kontinu atau diskrit, seperti skor pada skala (misalnya, 1-5 bintang, atau 1-10 poin). Dalam beberapa kasus, peringkatnya mungkin biner, yang menunjukkan preferensi “ya/tidak” atau “positif/negatif”.

- Peringkat Berdasarkan Daftar. Pakar manusia diminta untuk memberi peringkat atau mengurutkan kumpulan kemungkinan yang diberikan keluaran.

Di sini kami mempertimbangkan umpan balik perbandingan berpasangan karena ini adalah salah satu bentuk umpan balik manusia yang paling sederhana dan umum yang digunakan dalam RLHF. Dalam pengaturan ini, setiap kali, dua keluaran (y_a, y_b) diambil secara acak dari kumpulan kandidat $\{y_1, \dots, y_N\}$. Pakar manusia kemudian disajikan dengan pasangan ini dan diminta untuk memutuskan keluaran mana yang mereka suka berdasarkan kriteria tertentu, seperti kejelasan, relevansi, dan akurasi. Umpan balik manusia dapat dikodekan sebagai label biner, $y_a \succ y_b$ untuk preferensi untuk y_a , dan $y_b \succ y_a$ untuk preferensi untuk y_b .

Salah satu model yang sederhana dan banyak digunakan untuk menggambarkan perbandingan berpasangan tersebut adalah model Bradley-Terry [Bradley dan Terry, 1952]. Model ini merupakan model probabilistik yang memperkirakan probabilitas bahwa satu item lebih disukai daripada item lainnya. Dengan mengadaptasi model ini ke notasi yang digunakan di sini, kita dapat menulis probabilitas bahwa y_a lebih disukai daripada y_b dalam bentuk

$$\begin{aligned} \Pr(y_b \succ y_b|x) &= \frac{r(x,y)}{r(x,y) + e^{-r(x,y)}} \\ &= \frac{r(x,y)}{r(x,y) + r(x,y)} \\ &= \text{Sigmoid}(r(x, y) - r(x, yb)) \end{aligned} \quad (4.35)$$

Saat melatih model penghargaan, kami ingin memaksimalkan probabilitas preferensi ini. Fungsi kerugian berdasarkan model Bradley-Terry diberikan oleh

$$L_r(\hat{y}) = -E(x, y_a, y_b) \hat{y} \log \Pr(y_a \succ y_b|x) \quad (4.36)$$

di mana (x, y_a, y_b) diambil dari kumpulan data Dr yang dianotasi manusia yang terdiri dari pasangan preferensi keluaran dan masukan yang sesuai. \hat{y} mewakili parameter model penghargaan, yang mencakup parameter dekoder Transformer dan matriks pemetaan linier Wr . Dalam praktiknya, dengan asumsi (x, y_a, y_b) diambil sampelnya secara seragam dari Dr , kita dapat mengganti ekspektasi dengan penjumlahan

$$L_r(\hat{y}) = \hat{y} \frac{1}{|Dr|} \sum_{(x,y,yb) \in Dr} \log \Pr(y_b \succ y_b|x) \quad (4.37)$$

Tujuan dari pelatihan model reward adalah untuk menemukan parameter optimal \hat{y} yang meminimalkan hal ini. fungsi kerugian, diberikan oleh

$$\hat{y} = \arg \min_{\hat{y}} L_r(\hat{y}) \quad (4.38)$$

Karena model penghargaan itu sendiri juga merupakan LLM, kita dapat langsung menggunakan kembali prosedur pelatihan Transformer untuk mengoptimalkan model penghargaan. Perbedaan dari pelatihan LLM standar adalah kita hanya perlu mengganti kerugian entropi silang dengan kerugian perbandingan berpasangan seperti yang dijelaskan dalam Persamaan (4.37). Setelah pelatihan model penghargaan, kita dapat menerapkan model penghargaan terlatih $r(\cdot)$ untuk mengawasi LLM target agar selaras.

Perlu dicatat bahwa meskipun kami melatih model penghargaan untuk melakukan pemeringkatan berpasangan, kami menerapkannya untuk menilai setiap pasangan input-output secara independen selama proses penyelarasan.

tujuan pemeringkatan memastikan bahwa model penghargaan peka terhadap perbedaan kecil antara keluaran, tetapi kami mengandalkan skor berkelanjutan yang dihasilkan oleh model penghargaan untuk memandu pengoptimalan LLM. Keuntungan dari pendekatan ini adalah kami dapat memilih atau menggabungkan berbagai fungsi kerugian pemeringkatan, dan tetap menerapkan model penghargaan yang dihasilkan dengan cara yang sama seperti yang telah kami lakukan di subbagian ini. Konsistensi ini memastikan kerangka kerja terpadu untuk menyelaraskan LLM, terlepas dari kerugian pemeringkatan tertentu yang digunakan selama pelatihan model penghargaan.

4.3.3 Pelatihan LLM

Setelah memperoleh model imbalan, kami kemudian melatih kebijakan (yaitu, LLM) melalui metode A2C. Ingat kembali dari Bagian 4.3.1 bahwa urutan atau lintasan keadaan-tindakan \hat{y} dapat dievaluasi oleh fungsi utilitas

$$\text{Tentukan nilai } U(\hat{y} ; \hat{y}) = \log \hat{y} \hat{y}(\text{pada } /st)A(st, \text{ pada}) \quad (4.39)$$

tanggal = 1

di mana $A(st, at)$ adalah keuntungan mengambil tindakan *pada* keadaan *st* yang diberikan. Perkiraan $A(st, at)$ didefinisikan sebagai kesalahan $TD rt + \hat{y}V(st+1) - V(st)$, di mana fungsi nilai $V(st)$ dilatih dengan model hadiah.

Mengingat fungsi utilitas ini, fungsi kerugian berbasis A2C dapat ditulis dalam bentuk

$$\begin{aligned} L(\hat{y}) &= \hat{y}E\hat{y}\hat{y}D U(\hat{y} ; \hat{y}) \\ &= \hat{y}E\hat{y}\hat{y}D \log \hat{y} \hat{y}(\text{pada } /st)A(st, \text{ pada}) \end{aligned} \quad (4.40)$$

tanggal = 1

di mana D adalah ruang urutan keadaan-tindakan. Seperti biasa, tujuan pelatihan kebijakan adalah untuk meminimalkan fungsi kerugian ini.

$$\tilde{\hat{y}} = \arg \min_{\hat{y}} L(\hat{y}) \quad (4.41)$$

Jika kita memetakan masalah kembali ke masalah pemodelan bahasa dan mengadopsi notasi dari LLM, fungsi kerugian dapat ditulis sebagai:

$$L(\hat{y}) = \hat{y}E(x, y)\hat{y}D U(x, y; \hat{y}) \quad (4.42)$$

Di mana

$$\text{Tentukan nilai } U(x, y, \hat{y}) = \logaritma \hat{y} \hat{y}(yt | x, y < t)A(x, y < t, yt) \quad (4.43)$$

tanggal = 1

Di sini $\hat{y} \hat{y}(yt | x, y < t) = Pr(yt | x, y < t)$ adalah LLM yang diparameterisasi oleh \hat{y} .

Secara umum, kami tidak memiliki kumpulan data input-output beranotasi manusia D di RLHF, tetapi kumpulan data yang hanya berisi input. Output, dalam kasus ini, biasanya adalah prediksi yang dibuat oleh LLM.

Fungsi kerugian kemudian didefinisikan sebagai

$$L(\hat{y}) = \hat{y} \text{Ex}\hat{y} D\text{Ey}\hat{y}(\cdot|x) U(x, y; \hat{y}) \quad (4.44)$$

di mana D menunjukkan himpunan data input saja, dan $y \in \hat{y}(\cdot|x)$ menunjukkan bahwa keluaran y diambil sampelnya oleh kebijakan $\hat{y}(\cdot|x)$.

Rumusan di atas memberikan bentuk dasar metode A2C untuk LLM. Versi perbaikan dari model ini lebih umum digunakan dalam RLHF. Dalam pembahasan berikut, kita akan tetap menggunakan notasi pembelajaran penguatan untuk menyederhanakan penjelasan dan akan kembali menggunakan notasi pemodelan bahasa nanti.

Salah satu perbaikan umum dari metode gradien kebijakan adalah dengan menggunakan pengambilan sampel penting untuk menyempurnakan estimasi $U(\hat{y}; \hat{y})$. Hal ini dapat dituliskan sebagai

$$\text{Tentukan nilai } U(\hat{y}; \hat{y}) = \frac{\hat{y}(\text{pada } /st)}{\hat{y}\text{ref}(di /st)} A(st, di) \quad (4.45)$$

Di sini kita mengganti log-probabilitas log $\hat{y}(at/st)$ dengan rasio $\hat{y}(at/st) / \hat{y}\text{ref}(at/st)$.
Yer menunjukkan rasio parameter kebijakan sebelumnya (seperti model awal tempat kita memulai pelatihan). Jadi $\hat{y}(at/st)$ juga disebut fungsi rasio, dapat diartikan sebagai rasio log-probabilitas antara $\hat{y}\text{ref}(at/st)$, kebijakan saat ini \hat{y} dan kebijakan sebelumnya $\hat{y}\text{ref}$ (sebut saja kebijakan referensi). Dengan menggunakan fungsi rasio, kita memberi bobot ulang pada imbalan yang diamati berdasarkan kemungkinan tindakan di bawah > 1 , tindakan kebijakan saat ini dibandingkan dengan kebijakan acuan. Ketika $\hat{y}(at/st) / \hat{y}\text{ref}(at/st) < 1$, tindakan $\hat{y}\text{ref}(at/st)$ atau st kebijakan saat ini dibandingkan dengan kebijakan acuan. Sebaliknya, ketika $\hat{y}(at/st) / \hat{y}\text{ref}(at/st) < 1$, tindakan $\hat{y}\text{ref}(at/st)$ kurang disukai oleh kebijakan saat ini.

4Pertimbangkan kasus yang lebih umum di mana kita ingin mengevaluasi kebijakan menggunakan imbalan yang diharapkan (lihat juga Persamaan (4.18)).

$$J(\hat{y}) = E\hat{y}\hat{y} R(\hat{y}) \quad (4.46)$$

Di sini $\hat{y} \in \hat{y}$ berarti bahwa barisan \hat{y} dihasilkan oleh kebijakan \hat{y} . Atau, kita dapat menulis $J(\hat{y})$ dalam bentuk lain

$$J(\hat{y}) = E\hat{y}\hat{y}\hat{y}\text{ref} \quad \frac{Pr\hat{y}(\hat{y})}{Pra-ref(\hat{y})} \quad \begin{matrix} \text{Rumus } R(\hat{y}) \\ \text{Rumus } R(\hat{y}) \end{matrix} \quad (4.47)$$

Tidak sulit untuk menemukan bahwa ruas kanan persamaan ini pada dasarnya sama karena $(\hat{y})R(\hat{y}) = Pr\hat{y}(\hat{y})R(\hat{y}) = E\hat{y}\hat{y} R(\hat{y})$

E $\hat{y}\hat{y}\hat{y}\text{ref}$ $\frac{Pr\hat{y}(\hat{y})}{Pratinjau}$ Rumus untuk persamaan R adalah: Pra-ref (\hat{y}) $\frac{Pr\hat{y}(\hat{y})}{Pratinjau}$

Perhatikan bahwa kesetaraan ini hanya berlaku ketika ekspektasi dilakukan di seluruh ruang sekuen. Namun, dalam praktiknya, kita sering kali hanya mengambil sampel sejumlah kecil sekuen menggunakan kebijakan dalam pembelajaran kebijakan. Akibatnya, metode pengambilan sampel itu sendiri penting. Persamaan (4.47) menawarkan cara yang menarik untuk memisahkan proses perhitungan pengambilan sampel dan imbalan: pertama-tama kita menggunakan kebijakan dasar (dengan $\hat{y}\text{ref}$) untuk mengambil sampel sejumlah sekuen, dan kemudian menggunakan kebijakan target (dengan \hat{y}) untuk menghitung imbalan yang diharapkan. Dengan cara ini, kita memisahkan kebijakan yang digunakan untuk mengumpulkan data, dan kebijakan yang digunakan untuk menghitung gradien. Pendekatan ini menghindari kebutuhan untuk mengambil sampel langsung dari kebijakan yang kita evaluasi, yang dapat bermanfaat dalam kasus-kasus di mana

Dalam pembelajaran penguatan, $E\hat{y}\hat{y}\hat{y}\text{ref} / (\hat{y})R(\hat{y})$ sering disebut sebagai tujuan penggantian.

Persamaan $Pr\hat{y}(\hat{y})$ juga dapat diinterpretasikan dari perspektif gradien kebijakan. Untuk $E\hat{y}\hat{y}\hat{y}\text{ref} / (\hat{y})R(\hat{y})$, gradien pada $\hat{y} = \hat{y}\text{ref}$ diberikan oleh

$$\frac{\hat{y}}{\hat{y}E\hat{y}\hat{y}\hat{y}\text{ref}} \quad \frac{Pr\hat{y}(\hat{y})}{Pra-ref(\hat{y})} \quad \begin{matrix} \text{Rumus } R(\hat{y}) \\ \hat{y} = \hat{y} \end{matrix} \quad = E\hat{y}\hat{y}\hat{y}\text{ref} \quad \frac{\hat{y}Pr\hat{y}(\hat{y}) / \hat{y} = \hat{y}}{\hat{y}\hat{y}} \quad \begin{matrix} \text{Rumus } R(\hat{y}) \\ \hat{y} = \hat{y} \end{matrix} \quad (4.48)$$

Sisi kanan adalah bentuk standar yang digunakan dalam metode gradien kebijakan, yang berarti kita menghitung arah

Masalah dengan model yang disajikan dalam Persamaan (4.47) (dan juga dalam Persamaan (4.39)) adalah bahwa varians dalam estimasi gradien sering kali tinggi, yang membuat proses pembelajaran menjadi tidak stabil. Untuk mengurangi masalah ini, teknik seperti clipping sering digunakan untuk membatasi bobot kepentingan dan mencegah pembaruan yang besar. Versi clipping dari fungsi utilitas (juga disebut fungsi tujuan pengganti clipping) diberikan oleh

$$\text{Uclip}(\hat{y}; \hat{y}) = \text{Klip}_{\frac{\hat{y}\hat{y}(\text{pada } /st)}{\hat{y}\hat{y}\text{ref}(\text{pada } /st)}}^T A(st, di) \quad (4.49)$$

$$\text{Klip}_{\frac{\hat{y}\hat{y}(\text{pada } /st)}{\hat{y}\hat{y}\text{ref}(di /st)}} = \text{menit}_{\frac{\hat{y}\hat{y}(\text{pada } /st)}{\hat{y}\hat{y}\text{ref}(\text{pada } /st)}} st_{\frac{\hat{y}\hat{y}(\text{pada } /st)}{\hat{y}\hat{y}\text{ref}(\text{pada } /st)}} \text{terikat}_{\frac{\hat{y}\hat{y}(\text{pada } /st)}{\hat{y}\hat{y}\text{ref}(\text{pada } /st)}} 1 \leq \hat{y}, 1 + \hat{y} \quad (4.50)$$

Di sini fungsi $\text{bound}(1 \leq \hat{y}, 1 + \hat{y})$ membatasi fungsi rasio ke rentang $[1 \leq \hat{y}\hat{y}\text{ref}(\text{at|st}), \hat{y}, 1 + \hat{y}]$.

Peningkatan lebih lanjut terhadap model di atas adalah mempertimbangkan wilayah kepercayaan dalam optimasi [Schulman et al., 2015]. Dalam pembelajaran penguatan, pembaruan besar pada kebijakan dapat menyebabkan ketidakstabilan, di mana agen dapat mulai berkinerja lebih buruk setelah pembaruan. Ide yang masuk akal adalah mengoptimalkan model di wilayah kepercayaan, yang merujuk ke wilayah di sekitar estimasi parameter saat ini tempat model berperilaku baik. Salah satu pendekatan untuk menggabungkan wilayah kepercayaan adalah dengan memberlakukan batasan pada ukuran pembaruan kebijakan, memastikan bahwa kebijakan saat ini tidak menyimpang terlalu signifikan dari kebijakan referensi. Ini dapat dicapai dengan menambahkan penalti berdasarkan beberapa bentuk divergensi antara kebijakan saat ini dan referensi ke fungsi objektif. Bentuk sederhana dari penalti tersebut diberikan oleh perbedaan dalam log-probabilitas urutan \hat{y} di bawah kebijakan saat ini versus kebijakan referensi:

$$\text{Hukuman} = \log \hat{y}(\hat{y}) - \log \hat{y}\hat{y}\text{ref}(\hat{y}) \quad (4.51)$$

Dalam praktiknya, penalti ini dapat diperkirakan dengan hanya mempertimbangkan probabilitas kebijakan dan mengabaikan dinamikanya. Hal ini memberikan

$$\text{Penalti} = \text{log}_{\frac{\hat{y}\hat{y}(\text{pada } /st)}{\hat{y}\hat{y}\text{ref}(\text{pada } /st)}} \hat{y} - \log_{\frac{\hat{y}\hat{y}(\text{pada } /st)}{\hat{y}\hat{y}\text{ref}(\text{pada } /st)}} \hat{y}\hat{y}\text{ref}(\text{pada } /st) \quad (4.52)$$

Dengan menyertakan penalti ini dalam tujuan pengoptimalan, kami mendorong kebijakan saat ini agar tetap dekat dengan kebijakan referensi, membatasi pembaruan yang sangat besar yang dapat mengganggu stabilitas pembelajaran.

Kita dapat memasukkan penalti ini ke dalam fungsi tujuan pengganti yang terpotong, dan memperoleh

$$\text{Uppo-clip}(\hat{y}; \hat{y}) = \text{Uclip}(\hat{y}; \hat{y}) - \hat{y}\hat{y}\text{Penalti} \quad (4.53)$$

di mana \hat{y} adalah bobot penalti. Metode pelatihan ini disebut optimasi kebijakan proksimal (PPO), yang merupakan salah satu metode pembelajaran penguatan paling populer yang digunakan dalam LLM dan banyak bidang lainnya [Schulman et al., 2017].

Sekarang kita dapat menuliskan tujuan pelatihan LLM dalam bentuk PPO.

pembaruan parameter pada titik $\hat{y} = \hat{y}\hat{y}\text{ref}$ pada permukaan optimasi.

$$U(x, y; \hat{y}) = U_{\text{upper-clip}}(x, y; \hat{y}) - \hat{y}_{\text{Penalty}} \quad (4.54)$$

Di mana

$$U_{\text{upper-clip}}(x, y; \hat{y}) = \text{Klip}_{\frac{\hat{y}y(y|x, y < t)}{\hat{y}y_{\text{ref}}(y|x, y < t)}} \text{Sebuah}(x, y < t, yt) \quad (4.55)$$

$$\begin{aligned} \text{Penalty} &= \log Pr(y|x) - \log Pr_{\text{ref}}(y|x) \\ &= \logit Pr(yt|x, y < t) - \logit Pr_{\text{ref}}(yt|x, y < t) \end{aligned} \quad (4.56)$$

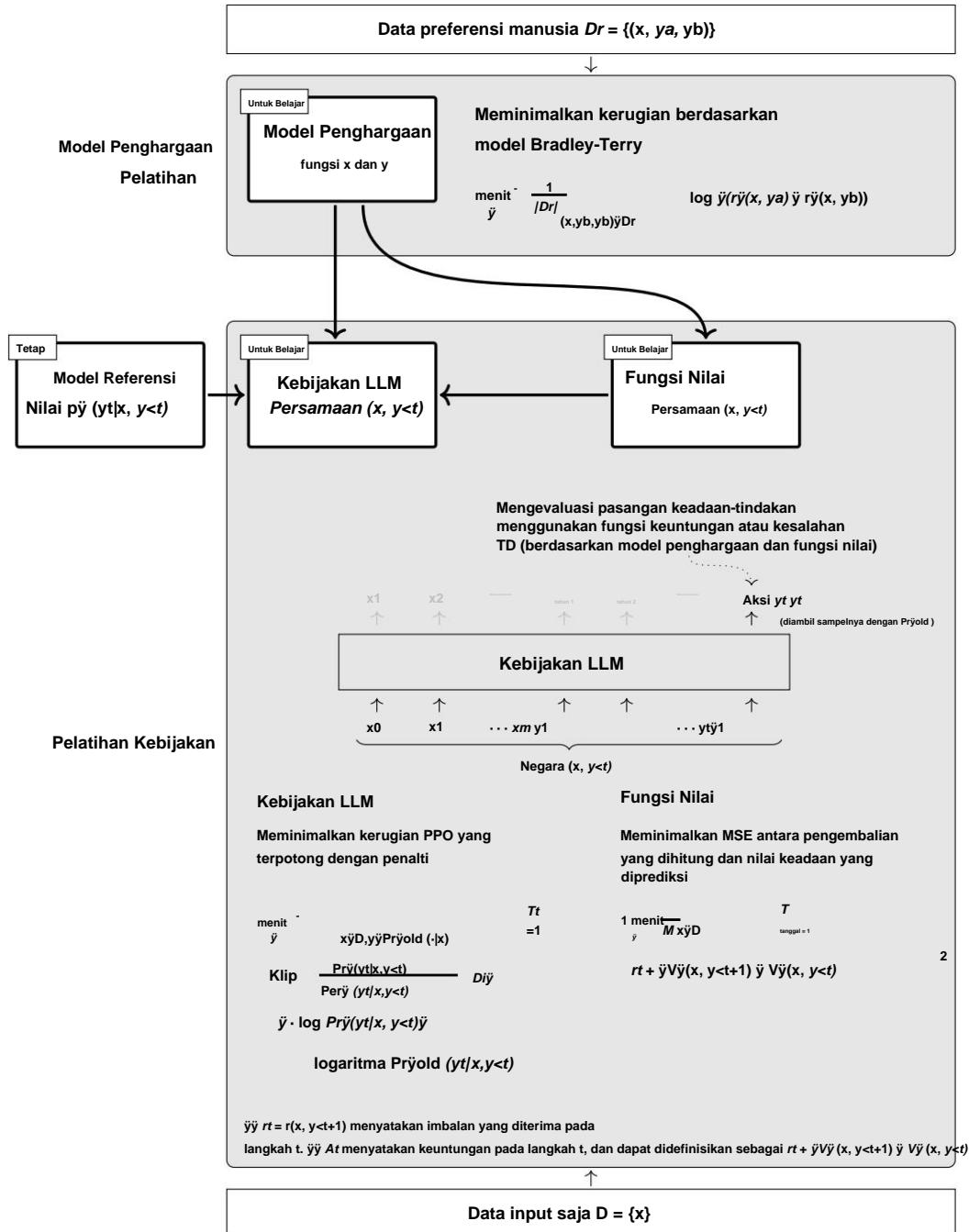
Meskipun notasi di sini tampak agak membosankan, gagasan PPO sederhana: kami mengembangkan tujuan dengan menggabungkan rasio kemungkinan terpotong dari kebijakan target dan referensi dengan fungsi keuntungan, lalu mengenakan penalti yang memastikan pembaruan kebijakan tidak terlalu besar.

RLHF berbasis PPO diilustrasikan dalam Gambar 4.9.

Singkatnya, penerapan RLHF memerlukan pembangunan empat model, yang semuanya berdasarkan Trans-arsitektur dekoder sebelumnya.

- Model Penghargaan ($r(y)$ di mana y menunjukkan parameter). Model penghargaan mempelajari data preferensi manusia untuk memprediksi penghargaan untuk setiap pasangan urutan token input dan output. Model ini merupakan dekoder Transformer yang diikuti oleh lapisan linier yang memetakan urutan (gabungan input dan output) ke skor penghargaan bernilai riil.
- Model Nilai atau Fungsi Nilai ($V(y)$ di mana y menunjukkan parameter). Fungsi nilai menerima skor imbalan dari model imbalan dan dilatih untuk memprediksi jumlah imbalan yang diharapkan yang dapat diperoleh mulai dari suatu keadaan. Fungsi ini umumnya didasarkan pada arsitektur yang sama dengan model imbalan.
- Model Referensi ($\hat{y}_{\text{ref}}(y)$ = $Pr_{\text{ref}}(y)$ di mana y menunjukkan parameter). Model referensi adalah LLM dasar yang berfungsi sebagai titik awal untuk pelatihan kebijakan. Dalam RLHF, model ini mewakili versi model sebelumnya atau model yang dilatih tanpa umpan balik manusia. Model ini digunakan untuk melakukan pengambilan sampel pada ruang keluaran dan berkontribusi pada perhitungan kerugian untuk pelatihan kebijakan.
- Model atau Kebijakan Target ($\hat{y}(y)$ = $Pr(y)$ di mana y menunjukkan parameter). Kebijakan ini mengatur bagaimana LLM memutuskan token berikutnya yang paling tepat berdasarkan konteksnya. Kebijakan ini dilatih di bawah pengawasan model hadiah dan model nilai.

Dalam praktiknya, model-model ini perlu dilatih dalam urutan tertentu. Pertama, kita perlu menginisialisasinya menggunakan beberapa model lain. Misalnya, model penghargaan dan model nilai dapat diinisialisasi dengan LLM yang telah dilatih sebelumnya, sementara model referensi dan model target dapat diinisialisasi dengan model yang telah disempurnakan instrukturnya. Perhatikan bahwa, pada titik ini, model referensi siap digunakan dan tidak akan diperbarui lebih lanjut. Kedua, kita perlu mengumpulkan data preferensi manusia dan melatih model penghargaan pada data ini. Ketiga, baik model nilai maupun kebijakan dilatih secara bersamaan menggunakan model penghargaan. Pada setiap posisi dalam urutan token keluaran, kita memperbarui m



Gambar 4.9: Ilustrasi RLHF. Langkah pertama adalah mengumpulkan data preferensi manusia dan melatih model imbalan menggunakan data ini. Setelah model imbalan dioptimalkan, bersama dengan model referensi, kami melanjutkan untuk melatih kebijakan dan fungsi nilai. Pada setiap langkah prediksi, kami menghitung jumlah kerugian berbasis PPO dan memperbarui parameter kebijakan. Ini memerlukan akses ke model imbalan, model referensi, dan fungsi nilai yang ada. Pada saat yang sama, kami memperbarui parameter fungsi nilai dengan meminimalkan kerugian MSE.

dengan meminimalkan kesalahan MSE dalam prediksi nilai, dan kebijakan diperbarui dengan meminimalkan kerugian PPO.

4.4 Peningkatan Penyelarasan Preferensi Manusia

Pada bagian sebelumnya, kita telah meninjau konsep dasar pembelajaran penguatan dan kerangka umum RLHF. Pada bagian ini, kita akan membahas beberapa penyempurnaan RLHF dan metode alternatif untuk mencapai penyelarasan preferensi manusia.

4.4.1 Pemodelan Penghargaan yang Lebih Baik

Pada Bagian 4.3.2, kami menyoroti tugas pembelajaran dari preferensi manusia serta penggunaan kerugian pemeringkatan berpasangan untuk melatih model penghargaan. Di sini kami mempertimbangkan lebih banyak metode untuk pemodelan penghargaan. Pembahasan kami akan relatif umum, dan karena model penghargaan banyak digunakan dalam banyak masalah pembelajaran penguatan, akan mudah bagi kami untuk menerapkan metode yang dibahas di sini pada RLHF dan aplikasi terkait.

4.4.1.1 Sinyal Pengawasan

Pelatihan model penghargaan secara umum dapat dilihat sebagai masalah pemeringkatan, di mana model belajar untuk memberikan skor pada keluaran sehingga urutannya mencerminkan preferensi yang ditunjukkan oleh manusia. Ada beberapa metode untuk melatih model penghargaan dari perspektif pemeringkatan.

Salah satu pendekatannya adalah memperluas pemeringkatan berpasangan ke pemeringkatan berdasarkan daftar. Untuk setiap sampel dalam satu set data, kita dapat menggunakan LLM untuk menghasilkan beberapa keluaran, dan meminta pakar manusia untuk mengurutkan keluaran ini. Misalnya, jika diberikan satu set yang terdiri dari empat keluaran $\{y_1, y_2, y_3, y_4\}$, salah satu urutannya adalah $y_2 \succ y_3 \succ y_1 \succ y_4$. Metode yang sangat sederhana untuk memodelkan urutan daftar adalah dengan mengakumulasikan kerugian perbandingan berpasangan. Misalnya, kita dapat mendefinisikan kerugian perbandingan berpasangan dengan mengakumulasikan kerugian pada semua pasangan keluaran:

$$\text{Daftar} = \sum_{(y_a, y_b) \in Y^2} \frac{\log \Pr(y_b | y_a)}{N(N-1)} \quad (4.57)$$

di mana Y adalah daftar keluaran, dan N adalah jumlah keluaran dalam daftar tersebut. $\Pr(y_a | y_b)$ dapat didefinisikan menggunakan model Bradley-Terry, yaitu, $\Pr(y_a | y_b) = \text{Sigmoid}(r(x, y_a) - r(x, y_b))$.

Di sini kita menghilangkan superskrip $\hat{\cdot}$ pada $\Pr(\cdot)$ agar notasi tetap rapi.

Perluasan model Bradley-Terry untuk pemeringkatan berdasarkan daftar dapat melibatkan mekanisme pemeringkatan yang memperhitungkan seluruh daftar keluaran, bukan hanya perbandingan berpasangan. Salah satu model tersebut adalah model Plackett-Luce, yang menggeneralisasi model Bradley-Terry untuk menangani beberapa item dalam pemeringkatan [Plackett, 1975]. Dalam model Plackett-Luce, untuk setiap item dalam daftar, kami mendefinisikan "nilai" untuk item ini yang mencerminkan kekuatan relatifnya untuk dipilih dibandingkan dengan item lainnya. Untuk masalah pemodelan hadiah di sini, nilai y dalam daftar Y dapat didefinisikan sebagai

$$\hat{y}(y) = \exp(r(x, y)) \quad (4.58)$$

Maka peluang memilih y dari Y diberikan oleh

$$\Pr(y \text{ dipilih} | x, Y) = \frac{\hat{y}(y)}{\sum_{y' \in Y} \hat{y}(y')} = \frac{\exp(r(x, y))}{\sum_{y' \in Y} \exp(r(x, y'))} \quad (4.59)$$

Misalkan Y merupakan suatu daftar berurut $y_1 \dot{y} y_2 \dot{y} \dots \dot{y} y_N$. Log-probabilitas keseluruhan dari daftar berurutan ini dapat didefinisikan sebagai jumlah log-probabilitas bersyarat pada setiap tahap pemilihan, yang diberikan oleh

$$\begin{aligned} \log \Pr(Y|X) &= \log \Pr(y_1 \dot{y} y_2 \dot{y} \dots \dot{y} y_N | X) = \log \Pr(y_1 | X, \\ &\quad \{y_2, \dots, y_N\}) + \log \Pr(y_2 | X, \{y_1, \dots, \\ &\quad y_N\}) + \\ &\quad \dots + \\ &= \sum_{k=1}^N \log \Pr(y_k | X, \{y_1, \dots, y_{k-1}\}) \end{aligned} \quad (4.60)$$

di mana $Y \setminus y_k$ merupakan subset dari daftar keluaran yang tidak dipilih pada tahap ke- k , yaitu $Y \setminus y_k = \{y_1, \dots, y_{k-1}\}$. Dengan log-probabilitas log $\Pr(Y|X)$, kita dapat mendefinisikan fungsi kerugian berdasarkan model Plackett-Luce dengan

$$L_{PL} = \sum_{y \in Y} E(x, y) \sum_{y' \in Y \setminus y} r(y, y') \log \Pr(y|X) \quad (4.61)$$

Ada juga banyak metode berpasangan dan metode daftar lainnya untuk memodelkan peringkat, seperti RankNet [Burges et al., 2005] dan ListNet [Cao et al., 2007]. Semua metode ini dapat dikategorikan ke dalam keluarga besar pendekatan pembelajaran untuk memberi peringkat, dan sebagian besarnya dapat diterapkan pada masalah pemodelan preferensi manusia. Akan tetapi, pembahasan metode ini berada di luar cakupan bab ini. Pembaca yang tertarik dapat merujuk ke buku-buku tentang topik ini untuk keterangan lebih rinci [Liu, 2009; Li, 2011].

Selain pemeringkatan berpasangan dan berdasarkan daftar, penggunaan metode pointwise untuk melatih model penghargaan menawarkan cara alternatif untuk menangkap preferensi manusia. Tidak seperti metode yang berfokus pada pemeringkatan relatif antara keluaran yang berbeda, metode pointwise memperlakukan setiap keluaran secara independen. Misalnya, pakar manusia mungkin menetapkan skor untuk keluaran individual, seperti peringkat pada skala lima poin. Tujuannya adalah untuk menyesuaikan model penghargaan sehingga keluarannya selaras dengan skor ini. Cara sederhana untuk mencapai pelatihan pointwise adalah melalui teknik regresi di mana penghargaan setiap keluaran diperlakukan sebagai variabel target. Misalkan $r(x, y)$ adalah skor yang diberikan untuk y yang diberikan x oleh manusia. Model penghargaan pointwise dapat dilatih dengan meminimalkan fungsi kerugian, sering kali berdasarkan kesalahan kuadrat rata-rata atau kerugian regresi lainnya, antara hadiah yang diprediksi $r(x, y)$ dan umpan balik manusia yang sebenarnya $y(x, y)$. Mis-

$$L_{\text{titik}} = \sum_{y \in Y} \sum_{y' \in Y \setminus y} r(y, y') \log \Pr(y|X) \quad (4.62)$$

Meskipun metode pointwise secara konseptual lebih sederhana dan dapat secara langsung memandu model penghargaan

memprediksi skor, metode ini mungkin tidak selalu menjadi pilihan terbaik dalam RLHF. Masalahnya adalah metode ini mungkin kesulitan dengan varians tinggi dalam umpan balik manusia, terutama ketika pakar yang berbeda memberikan skor yang tidak konsisten untuk keluaran yang serupa. Karena metode ini berfokus pada penyesuaian terhadap skor absolut daripada perbedaan relatif, ketidakkonsistenan dalam penilaian dapat menyebabkan kinerja model yang buruk. Selain itu, penyesuaian terhadap keluaran skor tertentu dapat menghambat generalisasi, terutama mengingat bahwa data pelatihan sering kali sangat terbatas dalam RLHF. Sebaliknya, metode yang mempertimbangkan preferensi relatif dapat mendorong pembelajaran pola keberhasilan dan kegagalan yang lebih umum. Meskipun demikian, ada skenario di mana metode pointwise mungkin masih cocok. Misalnya, dalam tugas-tugas di mana data pelatihan berlimpah dan biaya untuk memperoleh anotasi yang akurat dan konsisten rendah, metode pointwise dapat terbukti efektif.

Faktanya, untuk membuat sinyal supervisi untuk melatih model penghargaan lebih tangguh, kita juga dapat memperkenalkan istilah regularisasi tambahan ke dalam pelatihan. Misalnya, jika kita mempertimbangkan istilah pertama $\text{Uppo-clip}(x, y; \hat{y})$ dalam Persamaan (4.54) sebagai jenis penghargaan umum, maka istilah kedua (yaitu, istilah penalti) dapat dilihat sebagai bentuk regularisasi untuk model penghargaan, kecuali bahwa di sini tujuannya adalah untuk melatih kebijakan daripada model penghargaan. Contoh lain adalah [Eisenstein dkk. \[2023\]](#) mengembangkan istilah regularisasi berdasarkan jumlah kuadrat penghargaan, dan menambahkannya ke kerugian perbandingan berpasangan dalam RLHF:

$$\begin{aligned} L_{\text{reg}} &= L_{\text{pasangan}} + (\hat{y}E(x, ya, yb)\hat{y}Dr r(x, ya) + r(x, yb))^2 \\ &= \hat{y}E(x, y, yb)\hat{y}Dr \log Pr(\hat{y} yb|x) \\ &\quad \hat{y}E(x, y, yb)\hat{y}Dr r(x, y) + r(x, yb)^2 \end{aligned} \quad (4.63)$$

Mengoptimalkan dengan istilah regularisasi ini dapat membantu mengurangi kurangnya penentuan model imbalan⁵

4.4.1.2 Hadiyah yang Jarang vs. Hadiyah yang Padat

Seperti yang dibahas di Bagian 4.3, imbalan dalam RLHF sangat jarang: imbalan hanya diamati di akhir rangkaian, bukan terus-menerus selama proses pembangkitan. Berurusan dengan imbalan yang jarang telah lama menjadi perhatian dalam pembelajaran penguatan, dan telah menjadi salah satu tantangan dalam banyak aplikasi praktis. Misalnya, dalam robotika, sering kali perlu membentuk fungsi imbalan untuk memudahkan pengoptimalan daripada hanya mengandalkan imbalan di akhir rangkaian. Berbagai metode telah dikembangkan untuk mengatasi masalah ini. Salah satu pendekatan umum adalah pembentukan imbalan, di mana fungsi asli dimodifikasi untuk menyertakan imbalan antara, sehingga memberikan umpan balik yang lebih langsung. Selain itu, seseorang dapat mengadopsi pembelajaran kurikulum untuk menyusun tugas secara berurutan sedemikian rupa sehingga kompleksitasnya meningkat secara bertahap. Ini dapat membantu model untuk menguasai tugas yang lebih sederhana terlebih dahulu, yang mempersiapkan mereka untuk tantangan yang lebih kompleks saat keterampilan mereka berkembang. Ada banyak metode seperti itu yang dapat mengurangi dampak imbalan yang jarang, seperti metode Monte Carlo dan motivasi intrinsik. Sebagian besar metode ini bersifat umum dan pembahasannya dapat ditemukan dalam literatur yang

Meskipun kami tidak membahas metode untuk mengurangi sparse rewards secara rinci di sini, muncul pertanyaan menarik: mengapa sparse rewards begitu berhasil dalam RLHF? Ingat kembali dari Bagian 4.3.1 bahwa sinyal supervisi yang diterima pada setiap langkah waktu t bukanlah reward untuk tindakan saat ini, tetapi

⁵Model disebut kurang ditentukan jika ada beberapa set parameter alternatif yang dapat mencapai tujuan yang sama.

lebih tepatnya beberapa bentuk akumulasi imbalan dari t hingga langkah waktu terakhir. Sinyal supervisi tersebut padat di seluruh urutan, karena imbalan yang diperoleh di akhir urutan dapat ditransfer kembali ke langkah waktu tersebut, terlepas dari langkah waktu mana itu. Dengan kata lain, imbalan yang jarang diubah menjadi sinyal supervisi yang padat. Lebih jauh, dari perspektif pembentukan imbalan, Ng et al. [1999] menunjukkan bahwa imbalan pada t dapat didefinisikan sebagai

$${}' R_{\text{(tidak ada, pada}}, t+1} = r(t, \underset{\text{pada}}{f(x) + 1}) + {}' I(x) \underset{\text{pada}}{, st+1} \quad (4.64)$$

dimana $r'(\cdot)$ adalah fungsi imbalan yang telah ditransformasikan, $r(\cdot)$ adalah fungsi imbalan awal, dan $f(\cdot)$ adalah fungsi imbalan pembentuk. Untuk memastikan optimalitas kebijakan di bawah fungsi imbalan yang telah ditransformasikan, fungsi imbalan pembentuk dapat diberikan dalam bentuk

$$f_{\text{(keduanya, pada}}, st+1} = \bar{y} \bar{y}(st+1) \bar{y} \bar{y}(st) \quad (4.65)$$

dimana $\bar{y}(s)$ disebut nilai potensial dari keadaan s . Jika kita mendefinisikan $\bar{y}(s)$ sebagai fungsi nilai umum seperti pada persamaan (4.15) dan mensubstitusi persamaan (4.65) ke persamaan (4.64), kita memperoleh

$${}' R_{\text{(tidak ada, pada}}, t+1} = r(t, \underset{\text{pada}}{st+1}) + \bar{y} V(st+1) \bar{y} V(st) \quad (4.66)$$

Menarik untuk melihat bahwa fungsi ini persis sama dengan fungsi keuntungan yang digunakan dalam PPO. Hal ini menghubungkan metode berbasis keuntungan dengan pembentukan penghargaan: keuntungan pada hakikatnya adalah penghargaan yang dibentuk.

Di sisi lain, salah satu alasan untuk mengadopsi penghargaan akhir urutan terletak pada sifat tugas RLHF. Tidak seperti lingkungan pembelajaran penguatan tradisional tempat agen berinteraksi dengan lingkungan yang dinamis, tugas RLHF sering kali melibatkan pengambilan keputusan yang kompleks berdasarkan linguistik atau proses kognitif tingkat tinggi lainnya. Proses-proses ini tidak mudah menghasilkan penghargaan perantara yang sering dan bermakna karena kualitas dan kesesuaian tindakan hanya dapat dievaluasi sepenuhnya setelah mengamati dampaknya dalam konteks yang lebih luas dari keseluruhan urutan atau tugas. Dalam hal ini, sinyal penghargaan berdasarkan umpan balik manusia, meskipun sangat jarang, biasanya sangat informatif dan akurat. Akibatnya, kelangkaan ini, bersama dengan keinformatifan dan akurasi yang tinggi dari umpan balik manusia, dapat membuat pembelajaran menjadi kuat dan efisien.

4.4.1.3 Hadiah yang Berbutir Halus

Untuk banyak aplikasi, tujuan kita akan lebih kompleks daripada sekadar mengevaluasi keseluruhan teks. Misalnya, dalam analisis sentimen, kita sering kali tidak hanya menentukan sentimen suatu teks, tetapi perlu menganalisis sentimen tersebut secara lebih rinci dengan mengaitkannya dengan aspek-aspek tertentu dari suatu topik yang dibahas dalam teks tersebut. Perhatikan kalimat "Kamera ponsel ini sangat bagus, tetapi daya tahan baterainya mengecewakan." Dalam contoh ini, kita perlu menganalisis sentimen yang diungkapkan tentang kamera dan baterai secara terpisah. Analisis semacam itu, yang dikenal sebagai analisis sentimen berbasis aspek, membantu memberikan pemahaman yang lebih rinci tentang ulasan pelanggan dibandingkan dengan analisis sentimen umum.

Untuk masalah pemodelan hadiah, kita sering kali perlu memodelkan bagian-bagian berbeda dari suatu urutan juga. Cara yang sederhana dan mudah untuk melakukan ini adalah dengan membagi urutan menjadi beberapa segmen dan kemudian menghitung hadiah untuk setiap segmen [Wu et al., 2023]. Misalkan sebuah token keluaran

deret y dapat dibagi menjadi ns segmen $\{y^1, \dots, y^{ns}\}$ berdasarkan beberapa kriteria. Kita dapat menggunakan model penghargaan untuk mengevaluasi masing-masing segmen ini. Dengan mengambil x , y dan y^k sebagai input untuk penghargaan model, skor hadiah untuk segmen ke- k diberikan oleh

$$R_{\text{aw}} = r(x, y, y^k) \quad (4.67)$$

Kemudian skor hadiah untuk seluruh rangkaian keluaran diberikan oleh

$$\begin{aligned} & \text{Rumus untuk mencari nilai } x \text{ adalah} \\ & \sum_{k=1}^n r(x, y, y^k) \end{aligned} \quad (4.68)$$

Di sini $r(x, y)$ dapat digunakan untuk melatih kebijakan seperti biasa.

Masalah dengan model ini adalah bahwa pelatihan model penghargaan pada tingkat segmen tidak semudah mempelajari preferensi manusia pada keseluruhan teks, karena sulit untuk memperoleh model penghargaan tingkat segmen. data preferensi manusia. Untuk masalah seperti pemeringkatan (misalnya, kami menilai segmen menurut tingkatnya misinformasi), satu pendekatan sederhana adalah menetapkan skor peringkat untuk setiap segmen dan melatih model penghargaan menggunakan metode pointwise. Misalnya, kita dapat menggunakan LLM yang kuat untuk menilai urutan $y^1 \dots y^k y^1$ dan $y^1 \dots y^k$, dan memperoleh skor $s(y^1 \dots y^k y^1)$ dan $s(y^1 \dots y^k)$. Kita kemudian dapat tentukan skor segmen y^k sebagai selisih antara $s(y^1 \dots y^k)$ dan $s(y^1 \dots y^k y^1)$

$$\text{Rumus untuk mencari } s(y^k) = s(y^1 \dots y^k) - s(y^1 \dots y^k y^1) \quad (4.69)$$

Dengan menggunakan skor tingkat segmen ini, kita dapat melatih model hadiah dengan fungsi kerugian regresi.

$$\text{Nilai } L = \sum_{k=1}^n s(y^k) - r(x, y, y^k) \quad (4.70)$$

Terkadang, penyelarasan dapat diperlakukan sebagai masalah klasifikasi, misalnya, kami menilai apakah suatu segmen memiliki masalah etika. Dalam hal ini, segmen tersebut dapat diberi label etis atau tidak etis, baik oleh manusia atau menggunakan pengklasifikasi tambahan. Mengingat label segmen, kita dapat melatih model penghargaan menggunakan beberapa fungsi kerugian klasifikasi. Misalnya, anggaplah bahwa $r(x, y, y^k) = 1$ jika segmen tersebut diklasifikasikan sebagai tidak etis, dan $r(x, y, y^k) = 0$ jika tidak. Itu kehilangan engsel dari model klasifikasi biner pelatihan diberikan oleh

$$\text{Engsel} = \max(0, 1 - r(x, y, y^k)) \quad (4.71)$$

di mana $r^k \in \{0, 1\}$ menyatakan label kebenaran dasar.

Masalah yang tersisa di sini adalah bagaimana membagi y menjadi beberapa segmen. Salah satu pendekatannya adalah dengan mendefinisikan segmentasi dengan panjang tetap, di mana y dibagi menjadi potongan-potongan dengan panjang yang sama. Namun, hal ini mungkin tidak selalu menjadi ideal, karena isi urutannya mungkin tidak selaras dengan batasan yang ditetapkan. Alternatif Pendekatan ini adalah melakukan segmentasi berdasarkan isyarat linguistik atau semantik tertentu, seperti batasan kalimat, pergeseran topik, atau struktur bermakna lainnya dalam teks. Segmentasi semacam itu dapat dicapai dengan menggunakan sistem segmentasi linguistik atau mendorong LLM untuk mengidentifikasi jeda alami dalam urutan. Pendekatan lain adalah menggunakan metode segmentasi dinamis berdasarkan kompleksitas

⁶Untuk memungkinkan model penghargaan mengeluarkan kategori, kita dapat mengganti lapisan linier yang dijelaskan di Bagian 4.3.2 dengan Lapisan Softmax.

urutannya. Misalnya, segmen dapat didefinisikan jika terdapat perubahan signifikan dalam skor penghargaan, yang mungkin sesuai dengan perubahan dalam tugas yang dimodelkan.

4.4.1.4 Kombinasi Model Reward

Model penghargaan dapat dilihat sebagai proksi untuk lingkungan. Karena lingkungan yang sebenarnya sering kali terlalu rumit atau tidak diketahui, pengembangan proksi yang sempurna untuk lingkungan tersebut pada umumnya tidak mungkin dilakukan. Oleh karena itu, penyelarasan LLM yang berlebihan dengan proksi yang tidak sempurna ini dapat menyebabkan penurunan kinerja,⁷. Kita juga dapat dikenal sebagai masalah optimasi berlebihan [Stiennon et al., 2020; Gao et al., 2023a] menjelaskan hal ini melalui hukum Goodhart, yang menyatakan: ketika suatu ukuran menjadi target, maka ukuran tersebut tidak lagi baik [Goodhart, 1984].

Mengatasi masalah pengoptimalan berlebihan tidaklah mudah, dan belum ada solusi yang matang. Pendekatan yang ideal mungkin adalah mengembangkan model imbalan oracle yang dapat menangkap tujuan sebenarnya dari tugas dengan sempurna dan mencegah agen dari "menipu". Namun, membuat model seperti itu sangat sulit karena kompleksitas lingkungan dunia nyata, serta tantangan dalam mendefinisikan semua faktor relevan yang berkontribusi pada hasil yang diinginkan. Sebaliknya, pendekatan yang lebih praktis adalah menggabungkan beberapa model imbalan, sehingga mengurangi ketidakselarasan antara tujuan pelatihan dan tujuan sebenarnya yang muncul dari penggunaan satu model imbalan spesifik [Coste et al., 2024].

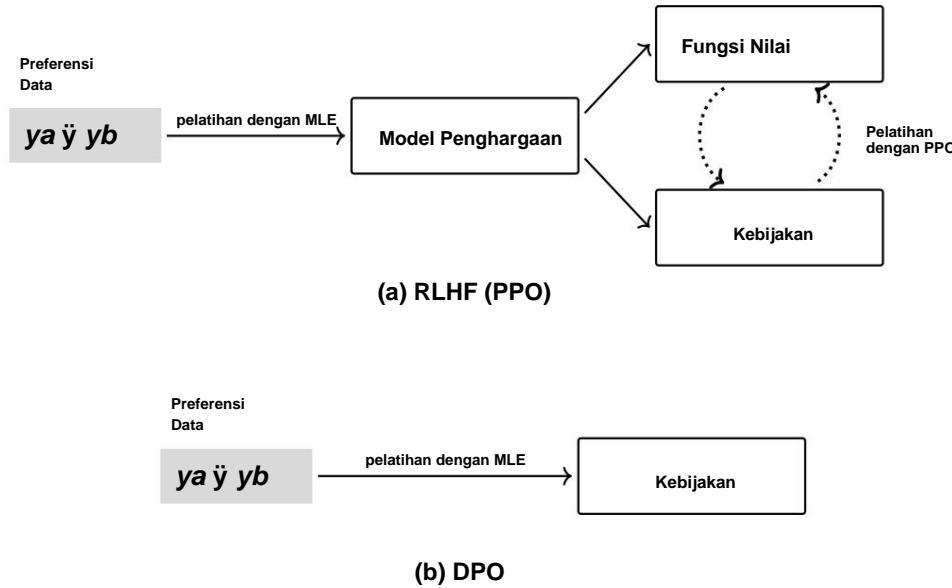
Mengingat sekumpulan model penghargaan, menggabungkannya adalah hal yang mudah, dan dalam beberapa kasus, kita dapat memperlakukan masalah ini sebagai masalah pembelajaran ensemble. Pendekatan yang sederhana namun umum adalah dengan merata-ratakan keluaran model-model ini untuk memperoleh estimasi penghargaan yang lebih tepat:

$$\text{menggabungkan} = \frac{1}{Nk} \sum_{=1}^N \text{minggu} \cdot \text{rk}(x, y) \quad (4.72)$$

di mana $\text{rk}(\cdot)$ adalah model imbalan ke- k dalam himpunan, w_k adalah bobot $\text{rk}(\cdot)$, dan K adalah jumlah model imbalan. Imbalan gabungan ini kemudian dapat digunakan untuk mengawasi pelatihan suatu kebijakan. Faktanya, ada banyak cara untuk menggabungkan berbagai model, misalnya, seseorang dapat membuat prediksi menggunakan rata-rata model Bayesian atau mengembangkan jaringan fusi untuk mempelajari cara menggabungkan prediksi dari berbagai model. Atau, seseorang dapat membungkai tugas ini sebagai masalah pengoptimalan multi-objektif, dan menggunakan beberapa model imbalan untuk melatih kebijakan secara bersamaan. Metode-metode ini telah dibahas secara intensif dalam literatur tentang pengoptimalan dan pembelajaran mesin [Miettinen, 1999; Bishop, 2006].

Selain metode kombinasi model, isu penting lainnya adalah bagaimana mengumpulkan atau membangun beberapa model penghargaan yang berbeda. Salah satu pendekatan yang paling sederhana adalah dengan menggunakan teknik pembelajaran ensemble, seperti mengembangkan berbagai model penghargaan dari berbagai subset dari kumpulan data tertentu atau dari berbagai sumber data. Untuk RLHF, juga memungkinkan untuk membangun model penghargaan berdasarkan pertimbangan berbagai aspek penyelarasan. Misalnya, kita dapat mengembangkan model penghargaan untuk mengevaluasi keakuratan faktual dari keluaran dan model penghargaan lain untuk mengevaluasi

⁷Masalah ini juga disebut reward hacking atau reward gaming [Krakovna et al., 2020; Skalse et al., 2022; Pan et al., 2022], yang merujuk pada fenomena saat agen mencoba mengelabui model hadiah tetapi gagal menyelaraskan tindakannya dengan tujuan sebenarnya dari tugas tersebut. Bayangkan seorang siswa yang diberi pekerjaan rumah dan diberi hadiah berupa poin atau pujiannya karena menyelesaikannya. Siswa tersebut kemudian mungkin menemukan cara untuk menyelesaikan pekerjaan rumah dengan upaya minimal untuk memaksimalkan hadiah, seperti menyalin dan menempelkan solusi dari internet atau tugas sebelumnya, daripada menyelesaikan masalahnya sendiri.



Gbr. 4.10: RLHF Standar (PPO) vs. DPO. Dalam RLHF, data preferensi manusia digunakan untuk melatih model imbalan, yang kemudian digunakan dalam pelatihan kebijakan serta fungsi nilai. Dalam DPO, penggunaan data preferensi manusia lebih langsung, dan kebijakan dilatih pada data ini tanpa perlu pelatihan model imbalan.

kelengkapan output. Kedua model ini saling melengkapi, dan dapat dikombinasikan untuk meningkatkan evaluasi output secara keseluruhan. Pendekatan lain adalah dengan menggunakan LLM siap pakai yang berbeda sebagai model penghargaan. Pendekatan ini sederhana dan praktis, karena telah ada banyak LLM yang dikembangkan dengan baik dan kita hanya perlu menggunakannya tanpa atau dengan sedikit modifikasi. Masalah yang menarik, meskipun tidak terkait erat dengan diskusi di sini, muncul: dapatkah LLM yang selaras dengan LLM lain mengungguli LLM tersebut? Mungkin tidak pada pandangan pertama. Sebagian, ini karena LLM target hanya meniru LLM lain berdasarkan supervisi terbatas dan dengan demikian tidak dapat menangkap dengan baik nuansa perilaku para supervisor ini. Namun, mengingat kemampuan generalisasi LLM yang kuat, pendekatan ini sebenarnya bisa sangat bermanfaat. Misalnya, penggunaan LLM sumber terbuka atau komersial sebagai model penghargaan telah menunjukkan kinerja yang kuat dalam menyelaraskan LLM, bahkan mencapai hasil terkini pada beberapa tugas populer [Lambert et al., 2024].

4.4.2 Optimasi Preferensi Langsung

Meskipun mempelajari model penghargaan merupakan langkah standar dalam pembelajaran penguatan, hal itu membuat seluruh proses pelatihan jauh lebih rumit daripada pelatihan yang diawasi. Melatih model penghargaan yang andal bukanlah tugas yang mudah, dan model penghargaan yang kurang terlatih dapat sangat memengaruhi hasil pembelajaran kebijakan. Kami sekarang mempertimbangkan metode penyelarasan alternatif, yang disebut pengoptimalan preferensi langsung (DPO), yang menyederhanakan kerangka pelatihan dengan menghilangkan kebutuhan untuk memodelkan penghargaan secara eksplisit [Rafailov et al., 2024]. Metode ini secara langsung mengoptimalkan kebijakan berdasarkan preferensi pengguna, daripada mengembangkan model penghargaan yang terpisah. Hasilnya, kita dapat mencapai penyelarasan preferensi manusia dengan cara seperti pembelajaran yang diawasi. Gambar 4.10 menunjukkan perbandingan metode RLHF standar dan metode DPO.

Sebelum menentukan tujuan DPO, mari kita tinjau terlebih dahulu tujuan pelatihan kebijakan yang digunakan dalam RLHF. Seperti yang dibahas di Bagian 4.3.3, kebijakan biasanya dilatih dengan mengoptimalkan fungsi kerugian.

dengan istilah penalti. Metode DPO mengasumsikan fungsi kerugian sederhana di mana kualitas output y yang diberikan input x dievaluasi oleh model penghargaan $r(x, y)$. Dengan demikian, tujuan pelatihan diberikan oleh

$$\hat{y} = \arg \min_{\hat{y}} \text{Contoh } DEy \hat{y} \hat{y}(-/x) \underbrace{r(x, y)}_{\text{kehilangan}} + \hat{y} (\log \hat{y} \hat{y}(y|x) - \log \hat{y} \hat{y} \text{ref}(y|x)) \underbrace{\text{penalti}}_{(4.73)}$$

Perhatikan bahwa dalam masalah optimasi ini, hanya istilah $\hat{y}(y|x)$ yang bergantung pada kebijakan target $\hat{y}(\cdot)$. Baik model imbalan $r(x, y)$ maupun model referensi $\hat{y} \hat{y} \text{ref}(y|x)$ diasumsikan tetap jika diberikan x dan y . Ini adalah asumsi yang kuat dibandingkan dengan PPO, tetapi seperti yang akan ditunjukkan nanti, ini menyederhanakan masalah dan penting untuk memperoleh tujuan DPO.

Karena \hat{y} adalah variabel yang ingin kita optimalkan, kita susun ulang ruas kanan persamaan (4.73) menjadi mengisolasi $\hat{y}(y|x)$ sebagai istilah independen:

$$\begin{aligned} \hat{y} = \arg \min_{\hat{y}} & \text{Contoh } DEy \hat{y} \hat{y}(-/x) \log \hat{y} \hat{y}(y|x) - \hat{y} \log \hat{y} \hat{y} \text{ref}(y|x) - r(x, y) \\ & \quad 1 \\ \text{Contoh } DEy \hat{y} \hat{y}(-/x) \log \hat{y} \hat{y}(y|x) - \log \hat{y} \hat{y} \text{ref}(y|x) + r(x, y) &= \arg \min_{\hat{y}} \hat{y} 1 \\ \hat{y} & \\ \text{Contoh } DEy \hat{y} \hat{y}(-/x) \log \hat{y} \hat{y}(y|x) - \log \hat{y} \hat{y} \text{ref}(y|x) \exp r(x, y) &= \arg \min_{\hat{y}} \hat{y} \\ & \quad \text{tergantung pada } \hat{y} \quad \text{tidak bergantung pada } \hat{y} \end{aligned} \quad (4.74)$$

Persamaan ini mendefinisikan fungsi objektif sebagai perbedaan antara fungsi distribusi log-probabilitas y dan fungsi y lainnya. Bentuk fungsi objektif ini tampaknya tidak “ideal”, karena kita biasanya lebih suka melihat perbedaan antara dua distribusi, sehingga kita dapat menafsirkan perbedaan ini sebagai semacam divergensi antara distribusi. Ide sederhananya adalah mengonversi suku kedua (yaitu, $\log \hat{y} \hat{y} \text{ref}(y|x) \exp(-r(x, y))$) ke dalam distribusi log-probabilitas di atas domain y . Jika kita memperlakukan $\hat{y} \hat{y} \text{ref}(y|x) \exp(-r(x, y))$ sebagai probabilitas tak ternormalisasi dari y , $\hat{y} \hat{y} \text{ref}(y|x) \exp(-r(x, y))$ dapat mengubahnya menjadi probabilitas ternormalisasi dengan membaginya dengan faktor normalisasi:

$$\text{Tentukan nilai } Z(x) \quad 1 \hat{y} \hat{y} \text{ref}(y|x) \exp r(x, y) \hat{y} \quad (4.75)$$

kamu

Oleh karena itu kita dapat mendefinisikan distribusi probabilitas dengan

$$\hat{y}^* (y|x) = \frac{\hat{y} \hat{y} \text{ref}(y|x) \exp r(x, y)}{Z(x)} \quad (4.76)$$

Bahasa Indonesia: Z(x)

Kita kemudian menulis ulang Persamaan (4.74) menjadi

$$\begin{aligned}
 &= \arg \min_{\tilde{y}} \frac{\tilde{y} \tilde{y}_{\text{ref}}(y|x) \exp^{-\tilde{y}}}{\text{persamaan } x \text{ dan } y} \\
 &\quad \text{Contoh: } DEy \tilde{y} \tilde{y}(-|x) \log \tilde{y}(y|x) \tilde{y} \log \\
 &\quad - \text{logaritma } Z(x) \\
 &\min_{\tilde{y}} \frac{\tilde{y} \tilde{y}_{\text{ref}}(y|x) \exp^{-\tilde{y}}}{\text{persamaan } x \text{ dan } y} \\
 &\quad \text{Contoh: } DEy \tilde{y} \tilde{y}(-|x) \log \tilde{y}(y|x) \tilde{y} \log \tilde{y} = \arg^* (y|x) \tilde{y} \log Z(x) \\
 &\min_{\tilde{y}} \frac{\tilde{y} \tilde{y}_{\text{ref}}(y|x) \exp^{-\tilde{y}}}{\text{persamaan } x \text{ dan } y} \\
 &\quad \text{Contoh: } D Ey \tilde{y} \tilde{y}(-|x) \log \tilde{y}(y|x) \tilde{y} \log \tilde{y} \tilde{y}(y|x) = \arg \\
 &\quad \tilde{y} \tilde{y}_{\text{ref}}(y|x) \log Z(x) \\
 &\arg \min_{\tilde{y}} \frac{\tilde{y} \tilde{y}_{\text{ref}}(y|x) \exp^{-\tilde{y}}}{\text{Divergensi KL}} \quad \text{Contoh: } D KL \tilde{y}(-|x) || \tilde{y} = \frac{(-|x)}{\text{konstan berkenaan dengan } \tilde{y}} - \text{logaritma } Z(x) \quad (4.77)
 \end{aligned}$$

Karena $\log Z(x)$ tidak bergantung pada \tilde{y} , maka hal itu tidak memengaruhi hasil operasi $\arg \min \tilde{y}$, dan dapat dihilangkan dari tujuan. Sekarang kita memperoleh tujuan pelatihan baru yang menemukan kebijakan optimal \tilde{y} dengan meminimalkan divergensi KL antara $\tilde{y}(-|x)$ dan $\tilde{y} \tilde{y}(-|x)$

$$\tilde{y} = \arg \min_{\tilde{y}} \frac{\tilde{y} \tilde{y}_{\text{ref}}(y|x) \exp^{-\tilde{y}}}{\text{Divergensi KL}} \quad (4.78)$$

Jelasnya, solusi untuk masalah optimasi ini diberikan oleh

$$\begin{aligned}
 &\text{Rumus } y|x = \tilde{y} \quad \frac{(-|x)}{\text{konstan berkenaan dengan } \tilde{y}} \\
 &= \frac{\tilde{y} \tilde{y}_{\text{ref}}(y|x) \exp^{-\tilde{y}}}{\text{persamaan } x \text{ dan } y} \quad (4.79)
 \end{aligned}$$

Mengingat persamaan ini, kita dapat mengekspresikan hadiah $r(x, y)$ menggunakan model target $\tilde{y} \tilde{y}(y|x)$, model referensi $\tilde{y} \tilde{y}_{\text{ref}}(y|x)$, dan faktor normalisasi $Z(x)$:

$$\begin{aligned}
 &\text{Rumus} \\
 &\text{untuk mencari } y / x \text{ adalah:} \quad (4.80)
 \end{aligned}$$

Hal ini menarik karena pada awalnya kita berusaha mempelajari kebijakan $\tilde{y}(-)$ menggunakan model imbalan $r(x, y)$, tetapi akhirnya memperoleh representasi model imbalan berdasarkan kebijakan tersebut. Mengingat model imbalan yang didefinisikan dalam Persamaan (4.80), kita dapat menerapkannya pada model Bradley-Terry untuk menghitung probabilitas preferensi (lihat juga Bagian 4.3.2):

$$\begin{aligned}
 \Pr(y \tilde{y} yb|x) &= \text{Sigmoid}(r(x, y) \tilde{y} r(x, yb)) \\
 &= \text{Sigmoid } \tilde{y} \log + \log Z(x) \frac{\tilde{y} \tilde{y}_{\text{ref}}(ya|x) \tilde{y} \tilde{y}(yb|x) + \log Z(x)}{\tilde{y} \log \tilde{y} \tilde{y}_{\text{ref}}(yb|x)} \\
 &\quad \frac{x) \tilde{y} \tilde{y}(ya/x)}{x) \tilde{y} \tilde{y}(yb/x)} \\
 &= \text{Sigmoid } \tilde{y} \log \tilde{y} \log \frac{\tilde{y} \tilde{y}_{\text{ref}}(ya|x) \tilde{y} \tilde{y}_{\text{ref}}(yb|x)}{———} \quad (4.81)
 \end{aligned}$$

Rumus ini elegan karena mengubah perbedaan imbalan menjadi perbedaan fungsi rasio, dan kita tidak perlu menghitung nilai $Z(x)$. Hasil langsungnya adalah kita tidak lagi memerlukan model imbalan, tetapi hanya memerlukan kebijakan target dan model referensi untuk menghitung probabilitas preferensi. Akhirnya, kita dapat melatih kebijakan target dengan meminimalkan fungsi kerugian DPO berikut

$$L_{DPO}(\hat{y}) = \int E(x, ya, yb) dR \log P(\hat{y} | ya, yb | x) \quad (4.82)$$

Bentuk fungsi kerugian ini sangat mirip dengan yang digunakan dalam pelatihan model imbalan di RLHF (lihat Persamaan (4.36)). Namun perlu dicatat bahwa fungsi kerugian di sini bergantung pada parameter kebijakan (yaitu, \hat{y}) dan bukan parameter model imbalan (yaitu, y).

Keunggulan utama DPO terletak pada kesederhanaan dan efisiensinya. Tujuan DPO sangat jelas — DPO secara langsung mengoptimalkan umpan balik berbasis preferensi, alih-alih mengandalkan model penghargaan yang dikembangkan secara terpisah. Selain itu, DPO secara umum lebih efisien dalam pengambilan sampel, karena DPO belajar dari kumpulan data tetap tanpa memerlukan proses pengambilan sampel yang mahal secara komputasi yang digunakan dalam PPO. Hal ini menjadikan DPO sebagai metode yang populer untuk penyelarasan preferensi manusia, terutama saat mengembangkan dan menerapkan model penghargaan melalui pembelajaran penguatan merupakan hal yang menantang.

DPO secara umum dapat dilihat sebagai metode pembelajaran penguatan luring, di mana data pelatihan telah dikumpulkan dan ditetapkan sebelumnya, dan tidak ada eksplorasi. Sebaliknya, metode pembelajaran penguatan daring seperti PPO, yang memerlukan eksplorasi status baru melalui interaksi dengan lingkungan (menggunakan model penghargaan sebagai proksi), juga memiliki keunggulan uniknya sendiri. Salah satu manfaat pembelajaran penguatan daring adalah memungkinkan agen untuk terus beradaptasi dengan perubahan lingkungan dengan belajar dari umpan balik waktu nyata. Ini berarti bahwa, tidak seperti metode luring, metode daring tidak dibatasi oleh sifat statis data yang telah dikumpulkan sebelumnya dan dapat menemukan strategi pemecahan masalah baru. Selain itu, eksplorasi dapat membantu agen mencakup rentang pasangan status-tindakan yang lebih luas, sehingga meningkatkan generalisasi. Ini bisa menjadi keuntungan penting bagi LLM, karena generalisasi dianggap sebagai aspek penting dalam menerapkan model besar tersebut.

4.4.3 Pembuatan Data Preferensi Otomatis

Meskipun belajar dari preferensi manusia merupakan metode yang efektif dan populer untuk menyelaraskan LLM, membuat anotasi pada data preferensi memerlukan biaya yang mahal. Penggunaan umpan balik manusia tidak hanya menghadapi masalah skalabilitas yang terbatas, tetapi juga dapat menimbulkan bias karena umpan balik manusia pada dasarnya bersifat subjektif. Hasilnya, seseorang dapat beralih ke metode umpan balik AI untuk mengatasi masalah skalabilitas dan konsistensi ini tanpa batasan yang terkait dengan pemberi anotasi manusia.

Seperti halnya pembuatan data untuk penyempurnaan instruksi, pembuatan data preferensi menggunakan LLM mudah dilakukan. Dengan sekumpulan masukan, pertama-tama kami menggunakan LLM untuk membuat pasangan keluaran. Kemudian, kami meminta LLM untuk memberi label preferensi di antara setiap pasangan keluaran, beserta masukannya yang sesuai. Berikut ini adalah contoh permintaan LLM untuk membuat label preferensi untuk sepasang respons layanan konsumen.

Pertimbangkan skenario layanan pelanggan saat pelanggan mengajukan permintaan. Anda akan meninjau dua respons terhadap permintaan ini. Harap nyatakan respons mana yang lebih disukai. Perhatikan bahwa respons yang baik harus sopan, jelas, dan ringkas. Respons harus langsung menanggapi masalah pelanggan, memberikan informasi bermanfaat atau solusi, dan mempertahankan nada positif.

Permintaan:

Halo, saya melihat bahwa pesanan saya belum sampai, padahal pesanan tersebut dijadwalkan akan sampai beberapa hari yang lalu. Bisakah Anda memberi tahu saya tentang statusnya? Terima kasih!

Respon A: Saya

mohon maaf atas keterlambatan ini dan memahami betapa mengecewakannya hal ini. Kami berusaha sebaik mungkin untuk menyelesaikan masalah ini dengan cepat untuk Anda.

Respon B: Hei,

hal-hal terjadi! Paket Anda akan sampai di sana saat sudah sampai, tidak perlu menekankan.

Respon A lebih disukai.

Setelah kami mengumpulkan label preferensi tersebut, kami dapat menggunakananya, bersama dengan pasangan keluaran dan masukan, untuk melatih model penghargaan. Tentu saja, kami dapat mempertimbangkan untuk menunjukkan beberapa contoh atau menggunakan teknik prompting tingkat lanjut, seperti CoT, untuk meningkatkan kinerja pelabelan. Misalnya, kami dapat menyertakan contoh yang menunjukkan bagaimana dan mengapa salah satu dari dua respons lebih disukai berdasarkan alasan CoT.

Selain label preferensi, kita juga dapat memperoleh probabilitas yang terkait dengan setiap label [Lee et al., 2023]. Metode sederhana adalah mengekstrak probabilitas untuk token label, seperti "A" dan "B", dari probabilitas yang dihasilkan oleh LLM. Kita kemudian dapat menggunakan fungsi Softmax atau teknik normalisasi lainnya untuk menormalkan kembali probabilitas ini ke dalam distribusi atas label.

Probabilitas label yang disukai ini dapat berfungsi sebagai sinyal pengawasan titik demi titik untuk melatih model penghargaan, seperti yang dibahas dalam Bagian 4.4.1.

Untuk pembuatan data, meskipun mudah untuk ditingkatkan, sering kali perlu memastikan data tersebut akurat dan beragam. Di sini, masalah kualitas dan keragaman data tidak hanya melibatkan pelabelan preferensi tetapi juga masukan dan keluaran model. Oleh karena itu, kita sering kali perlu menggunakan berbagai teknik untuk memperoleh data berkualitas tinggi dalam skala besar. Misalnya, seseorang dapat membuat keluaran dan anotasi model yang beragam dengan menggunakan LLM, perintah, demonstrasi dalam konteks yang berbeda, dan sebagainya [Cui et al., 2024]. Dubois et al. [2024] melaporkan bahwa variabilitas dalam data preferensi berpasangan penting untuk melatih LLM dari umpan balik manusia atau AI.

Meskipun pembelajaran dari umpan balik AI sangat terukur dan umumnya objektif, metode ini lebih cocok untuk tugas-tugas yang terdefinisi dengan baik di mana metrik kinerja objektif tersedia. Sebaliknya, pembelajaran dari umpan balik manusia lebih menguntungkan ketika menyelaraskan sistem AI dengan nilai-nilai manusia, preferensi, dan tugas-tugas dunia nyata yang kompleks yang memerlukan pemahaman konteks yang halus atau subjektif. Metode-metode ini dapat dikombinasikan untuk melatih LLM yang mendapatkan manfaat dari wawasan manusia dan skalabilitas umpan balik AI.

4.4.4 Penyelarasan Langkah demi Langkah

Sejauh ini, pembahasan kita tentang penyelarasan terutama difokuskan pada penggunaan model penghargaan untuk mengevaluasi seluruh pasangan urutan masukan-keluaran. Metode ini dapat dengan mudah diadaptasi ke skenario di mana kebenaran suatu keluaran dapat diperiksa dengan memeriksa apakah hasil yang diinginkan benar. Misalnya, dalam tugas menghitung ekspresi matematika, model penghargaan dapat memberikan umpan balik positif jika jawabannya benar, dan umpan balik negatif jika jawabannya salah. Namun, dalam banyak soal yang memerlukan penalaran yang rumit, sekadar memeriksa kebenaran hasil akhir tidaklah cukup untuk pembelajaran. Bayangkan seorang siswa yang hanya diberi jawaban akhir untuk soal matematika yang menantang. Mengetahui apakah jawaban akhir itu benar atau salah tidak membantu siswa mengetahui di mana kesalahannya dan cara menghitung jawaban yang benar. Pendekatan yang lebih baik adalah membimbing siswa dengan uraian langkah demi langkah dari proses pemecahan masalah dan mendorong pemahaman tentang konsep dan logika yang mendasari di balik langkah-langkah ini.

Bahasa Indonesia: Pada Bab 3, kami mempelajari metode CoT untuk mendorong LLM untuk secara eksplisit menuliskan langkah-langkah perantara atau proses penalaran yang diperlukan untuk mencapai kesimpulan atau memecahkan masalah. Kami melihat bahwa memecah masalah menjadi bagian-bagian yang lebih kecil dapat mempermudah pemahaman jalur solusi dan meningkatkan akurasi output. Metode-metode ini secara alami dapat diperluas ke penyelarasan LLM, yaitu, kami mengawasi model selama langkah-langkah perantara penalaran. Pertimbangkan tugas penalaran di mana LLM menghasilkan urutan langkah-langkah penalaran $y = \{y^1, \dots, y^{ns}\}$ untuk input yang diberikan. Hasil penalaran diasumsikan termasuk \dots dan bisa dalam langkah terakhir y^{ns} , dapat dengan mudah diverifikasi. Untuk masalah penalaran ini, [Uesato et al. \[2022\]](#) mengkategorikan pendekatan fine-tuning LLM ke dalam dua kelas:

- **Pendekatan Berbasis Hasil.** Supervisi hanya terjadi ketika hasil akhir diverifikasi. Ini adalah metode standar untuk belajar dari umpan balik manusia yang telah kita bahas dalam bab ini. Misalnya, LLM dioptimalkan untuk memaksimalkan beberapa bentuk imbalan $r(x, y)$.
- **Pendekatan Berbasis Proses.** Supervisi terlibat dalam semua langkah antara selain langkah terakhir. Untuk melakukan ini, kita perlu mengembangkan model untuk memberikan sinyal supervisi pada setiap langkah, dan mengembangkan fungsi kerugian yang dapat memanfaatkan sinyal supervisi tersebut.

Gambar 4.11 menunjukkan dua keluaran LLM untuk contoh soal matematika. Meskipun LLM memberikan jawaban akhir yang benar dalam kedua kasus, LLM membuat kesalahan selama proses penyelesaian masalah pada keluaran kedua. Pendekatan berbasis hasil mengabaikan kesalahan ini dan memberikan umpan balik positif untuk keseluruhan solusi. Sebaliknya, pendekatan berbasis proses dapat memperhitungkan kesalahan ini dan memberikan panduan tambahan pada langkah-langkah penalaran terperinci.

Masalah penting untuk pendekatan berbasis proses adalah bahwa kita perlu mendapatkan umpan balik tingkat langkah selama jalur penalaran yang (berpotensi) panjang. Kita dapat mengumpulkan atau membuat jalur penalaran yang sesuai dengan masalah dari kumpulan data yang ada. Pakar manusia kemudian membuat anotasi setiap langkah di jalur ini untuk mengetahui kebenarannya. Anotasi ini dapat digunakan untuk melatih LLMS secara langsung atau sebagai hadiah dalam pemodelan hadiah. Namun, dalam praktiknya, anotasi yang lebih kaya sering kali diperkenalkan [\[Lightman et al., 2024\]](#). Selain label yang benar dan salah, suatu langkah juga dapat diberi label netral untuk menunjukkan bahwa meskipun langkah tersebut mungkin secara teknis benar, langkah itu mungkin masih bermasalah dalam keseluruhan proses penalaran. Lebih jauh, untuk meningkatkan efisiensi anotasi data, teknik seperti pembelajaran aktif dapat digunakan. Mengidentifikasi kesalahan yang

Pertanyaan (dari Olimpiade Matematika Junior Internasional):

Jessica memiliki banyak aplikasi di ponselnya. Jumlah aplikasi yang dimilikinya dapat dibagi 5. Jika ia mengunduh 10 aplikasi lagi, ia akan tetap memiliki kurang dari 50 aplikasi. Berapa jumlah aplikasi maksimum yang dapat dimiliki Jessica?

Respon 1:

y⁻¹: x harus kelipatan 5, yaitu $x \equiv 0 \pmod{5}$.

Setelah mengunduh 10 aplikasi, total

y⁻²: jumlah total aplikasi seharusnya masih kurang dari 50: $x + 10 < 50$.

Menyelesaikan pertidaksamaan ini: $x < 40$.

y⁻³: Karena x harus kelipatan 5 dan kurang dari 40, nilai x terbesar adalah 35.

y⁻⁴: Verifikasi hasilnya. Jika $x = 35$, maka setelah mengunduh 10 aplikasi, Jessica akan memiliki: $35 + 10 = 45$ yang memenuhi $45 < 50$.

y⁻⁵: Jumlah aplikasi asli maksimum adalah 35.

↑
benar

Respon 2:

y⁻¹: Jumlah aplikasi habis dibagi 5, jadi x harus kelipatan 5.

y⁻²: Jika dia mengunduh 10 aplikasi lagi, total barunya akan menjadi $x + 10$, dan ini harus kurang dari 50. Oleh karena itu, $x + 10 < 50$.

Menyederhanakan ini menghasilkan $x < 40$.

y⁻³: x harus kurang dari atau sama dengan kelipatan 5 terbesar, yaitu 40. bermasalah

y⁻⁴: Tapi $x + 10$ seharusnya tidak lebih besar atau sama dengan 50. Jadi, kita perlu mengurangi 5 dari 40. bermasalah

y⁻⁵: Jadi, hasil akhirnya adalah 35.

↑
benar

Gbr. 4.11: Dua respons LLM terhadap soal matematika. Pada respons 1, hasil akhir dan semua langkah penalaran sudah benar. Pada respons 2, hasil akhir sudah benar, tetapi ada kesalahan dalam proses penalaran (disorot dengan warna merah). Untuk pendekatan berbasis hasil, kedua respons dianggap benar. Untuk pendekatan berbasis proses, kesalahan pada respons 2 dapat dipertimbangkan dalam pemodelan penghargaan.

belajar dari kesalahan penalaran. Sebaliknya, memberi anotasi pada langkah-langkah yang dianggap benar oleh model tetapi sebenarnya bermasalah seringkali lebih efektif.

Dengan serangkaian jalur penalaran beranotasi tingkat langkah dan masukan yang sesuai, kita dapat melatih model penghargaan untuk memberikan umpan balik guna mengawasi pembelajaran kebijakan. Model penghargaan dapat diperlakukan sebagai model klasifikasi, sehingga arsitekturnya dapat berupa dekoder Transformer dengan lapisan Softmax yang ditumpuk di atasnya. Pada langkah k , model penghargaan mengambil deskripsi masalah (dilambangkan dengan x) dan langkah penalaran yang dihasilkan sejauh ini (dilambangkan dengan $y^{\sim k}$) sebagai masukan dan menghasilkan distribusi probabilitas atas set label {benar, salah} atau {benar, salah, netral}. Kemudian model reward yang dipelajari digunakan untuk mengevaluasi jalur penalaran dengan menilai kebenaran setiap langkah. Metode sederhana untuk memodelkan kebenaran adalah dengan menghitung jumlah langkah yang diklasifikasikan sebagai benar, yang diberikan oleh

$$\sum_{k=1}^K \mathbb{I}(\text{benar}, C(x, y^{\sim k})) \quad (4.83)$$

di mana $C(x, y^{\sim k})$ menunjukkan label dengan probabilitas maksimum. Kita juga dapat menggunakan log-probabilitas

klasifikasi untuk menentukan hadiah dari seluruh jalur

$$\text{Rumus untuk mencari nilai } x \text{ adalah} \quad \log \Pr(\text{benar}|x, y^{\text{yk}}) \quad (4.84)$$

$$k=1$$

di mana $\Pr(\text{benar}|x, y^{\text{yk}})$ menunjukkan probabilitas label benar yang dihasilkan oleh model hadiah. Skor hadiah $r(x, y)$ kemudian dapat digunakan untuk melatih kebijakan dalam RLHF seperti biasa.

Meskipun kami membatasi pembahasan pada soal matematika, pendekatan yang dijelaskan di sini bersifat umum dan dapat diterapkan pada berbagai macam tugas yang melibatkan penalaran dan pengambilan keputusan multi-langkah. Selain itu, kami dapat mempertimbangkan berbagai aspek saat menilai kualitas suatu langkah, bukan hanya kebenarannya. Misalnya, dalam sistem dialog, respons tidak hanya harus akurat tetapi juga sesuai konteks di berbagai putaran percakapan. Jika suatu model memberikan respons yang benar tetapi gagal mempertahankan koherensi dalam konteks dialog yang sedang berlangsung, umpan balik tingkat langkah dapat membantu model mengidentifikasi dan mengoreksi ketidaksesuaian tersebut. Perhatikan juga bahwa pendekatan berbasis proses terkait dengan pendekatan pemodelan penghargaan terperinci yang dibahas dalam Bagian 4.4.1.3. Semua pendekatan ini pada dasarnya bertujuan untuk memberikan pengawasan yang lebih rinci kepada LLM dengan membagi output mereka menjadi langkah-langkah yang lebih kecil dan lebih mudah dikelola. Namun, umpan balik berbasis proses lebih berfokus pada evaluasi kebenaran suatu langkah berdasarkan langkah-langkah sebelumnya, sedangkan pendekatan dalam Bagian 4.4.1.3 menekankan evaluasi setiap langkah secara independen.

Gagasan menyelaraskan LLM langkah demi langkah memiliki potensi aplikasi yang besar, terutama mengingat pergeseran baru-baru ini ke arah tugas penalaran yang lebih kompleks dalam penggunaan LLM. Misalnya, model GPT-01 dan GPT-03 dirancang dengan teknik penalaran yang lebih maju (seperti CoT internal yang panjang) untuk memecahkan masalah yang menantang seperti penalaran ilmiah dan matematika [OpenAI, 2024]. Tugas-tugas ini sering kali bergantung pada jalur penalaran yang panjang dan kompleks, dan oleh karena itu, tampaknya penting untuk memperkenalkan sinyal supervisi terperinci dalam proses penalaran. Selain itu, dari perspektif praktis, supervisi yang efektif pada jalur penalaran yang panjang tidak hanya meningkatkan kinerja penalaran, tetapi juga membantu model menghilangkan langkah-langkah penalaran yang berlebihan atau tidak perlu, sehingga mengurangi kompleksitas penalaran dan meningkatkan efisiensi.

4.4.5 Penyelarasan Waktu Inferensi

Pada bagian ini kami mengeksplorasi berbagai metode untuk menyelaraskan model dengan preferensi dan anotasi manusia. Akan tetapi, salah satu keterbatasan signifikan dari banyak metode tersebut adalah bahwa LLM harus disetel dengan baik. Untuk RLHF dan variannya, melatih LLM dengan model imbalan dapat memakan biaya komputasi yang mahal dan tidak stabil, yang menyebabkan peningkatan kompleksitas dan biaya saat menerapkan pendekatan ini. Dalam kasus ini, kami dapat mempertimbangkan untuk menyelaraskan model pada waktu inferensi, sehingga menghindari kompleksitas dan upaya tambahan yang diperlukan.

Salah satu cara sederhana untuk mencapai penyelarasan inferensi-waktu adalah dengan menggunakan model penghargaan untuk memilih yang terbaik dari N keluaran alternatif yang dihasilkan oleh LLM, sebuah metode yang dikenal sebagai pengambilan sampel Best-of-N (pengambilan sampel BoN). Kita dapat menganggap pengambilan sampel BoN sebagai bentuk pemeringkatan ulang. Faktanya, metode pemeringkatan ulang telah banyak digunakan dalam tugas-tugas NLP, seperti penerjemahan mesin, untuk waktu yang lama. Metode ini biasanya diterapkan dalam situasi di mana pelatihan model yang kompleks membutuhkan biaya yang mahal. Dalam kasus seperti itu, pemeringkatan ulang keluaran secara langsung memungkinkan penggabungan model-model kompleks ini dengan biaya yang sangat rendah⁸

⁸Metode pemeringkatan ulang juga dapat membantu kita mengeksplorasi apa yang disebut sebagai kesalahan model dan kesalahan pencarian, meskipun hal ini

4.5 Ringkasan

Dalam proses pengambilan sampel BoN, LLM mengambil urutan input x dan menghasilkan N urutan yang berbeda. urutan keluaran $\{y^1, \dots, y^N\}$:

$$\{y^1, \dots, y^N\} = \text{argTopN} [\Pr(y|x)] \quad (4.85)$$

di mana operasi argTopN mengembalikan keluaran N teratas yang memaksimalkan fungsi $\Pr(y|x)$. Output ini dapat dihasilkan dengan berbagai cara, tergantung pada algoritma pencarian yang digunakan oleh model (misalnya, pencarian sampel atau sinar). Setelah kandidat output N -terbaik dihasilkan, model penghargaan digunakan untuk mengevaluasi dan memilih yang terbaik:

$$y^{\text{terbaik}} = \max\{r(x, y^1), \dots, r(x, y^N)\} \quad (4.86)$$

Perlu dicatat bahwa hasil pengambilan sampel BoN juga dipengaruhi oleh keragaman daftar N -terbaik. Ini adalah masalah umum dengan sebagian besar metode pemeringkatan ulang. Biasanya, kami ingin kandidat keluaran N -terbaik memiliki kualitas yang relatif tinggi tetapi cukup berbeda satu sama lain. Dalam banyak sistem pembangkitan teks, keluaran N -terbaik sangat mirip, sering kali hanya berbeda satu atau dua kata. Masalah keragaman bahkan lebih menantang dalam LLM, karena keluaran N -terbaik yang dihasilkan oleh LLM dapat berbeda dalam kata-katanya, tetapi makna semantiknya sering kali cukup mirip. Dalam praktiknya, seseorang dapat menyesuaikan hiperparameter model dan/atau mengadopsi LLM yang berbeda untuk menghasilkan kandidat keluaran yang lebih beragam untuk pemeringkatan ulang. Namun demikian, seperti halnya banyak sistem praktis, kita perlu membuat trade-off antara memilih kandidat berkualitas tinggi dan memastikan variasi yang cukup dalam keluaran yang dihasilkan.

Pengambilan sampel BoN juga dapat digunakan untuk melatih LLM. Metode yang terkait erat adalah pengambilan sampel penolakan. Dalam metode ini, pertama-tama kami memilih keluaran "terbaik" dari daftar N -terbaik melalui model penghargaan, lalu mengambil keluaran terpilih ini untuk menyempurnakan LLM. Dengan cara ini, kami dapat memperkenalkan preferensi manusia ke dalam pelatihan LLM melalui pendekatan yang jauh lebih sederhana dibandingkan dengan metode lain. Banyak LLM telah mengadopsi sampel penolakan untuk penyempurnaan [Nakano et al., 2021; Touvron et al., 2023b].

4.5 Ringkasan

Dalam bab ini, kami telah mengeksplorasi berbagai teknik untuk menyelaraskan LLM. Secara khusus, kami telah membahas metode fine-tuning yang memungkinkan LLM mengikuti instruksi dan menyelaraskannya dengan preferensi manusia. Salah satu manfaat fine-tuning LLM adalah efisiensi komputasi. Tidak seperti pra-pelatihan berdasarkan pengoptimalan jaringan saraf skala besar, fine-tuning adalah langkah pasca-pelatihan dan karenanya lebih murah secara komputasi. Selain itu, lebih cocok untuk mengatasi masalah yang tidak mudah dipecahkan dalam pra-pelatihan, seperti penyelarasan nilai manusia. Perhatian yang luas terhadap masalah penyelarasan juga telah menyebabkan lonjakan makalah penelitian tentang topik ini, yang telah menimbulkan tantangan dalam penulisan bab ini, karena sulit untuk mencakup semua teknik terbaru. Namun, kami telah mencoba memberikan pengantar yang relatif terperinci tentang pendekatan mendasar untuk penyelarasan, seperti

Masalah tidak sering dibahas dalam konteks LLM. Misalnya, anggaplah kita memiliki model lama dan model baru yang lebih canggih. Kita dapat menggunakan model baru untuk memilih keluaran terbaik dari daftar N -terbaik model lama sebagai keluaran oracle. Perbedaan kinerja antara keluaran oracle dan keluaran top-1 dari daftar N -terbaik asli mencerminkan peningkatan kinerja yang dibawa oleh model baru. Jika peningkatan kinerja signifikan, kita dapat mengatakan bahwa model lama memiliki lebih banyak kesalahan model. Jika peningkatannya kecil, ini mungkin menunjukkan bahwa masalahnya terletak pada kesalahan pencarian, karena kandidat terbaik tidak ditemukan.

instruksi fine-tuning dan RLHF.

Meskipun kami telah berfokus pada teknik penyelarasan LLM dalam bab ini, istilah penyelarasan AI merupakan konsep yang luas. Istilah ini secara umum merujuk pada proses untuk memastikan bahwa perilaku sistem AI selaras dengan nilai, tujuan, dan harapan manusia. Ide penyelarasan AI dapat ditelusuri kembali ke masa-masa awal AI. Deskripsi penyelarasan AI yang banyak dikutip berasal dari sebuah artikel oleh matematikawan dan ilmuwan komputer Norbert Wiener [[Wiener, 1960](#)]. Kutipannya adalah sebagai berikut

Jika kita menggunakan, untuk mencapai maksud-maksud kita, suatu badan mekanis yang pengoperasiannya tidak dapat kita campuri secara efisien... sebaiknya kita yakin sepenuhnya bahwa maksud yang dimasukkan ke dalam mesin itu adalah maksud yang sungguh-sungguh kita inginkan.

Saat itu, penyelarasan AI masih menjadi perhatian yang jauh bagi para peneliti. Namun, kini, hal itu sangat memengaruhi desain berbagai sistem AI. Misalnya, dalam robotika, penyelarasan sangat penting untuk memastikan bahwa robot otonom berinteraksi dengan aman dengan manusia dan lingkungannya. Dalam mengemudi secara otonom, mobil tidak hanya harus mematuhi peraturan lalu lintas, tetapi juga harus membuat keputusan yang rumit dan real-time yang mengutamakan keselamatan manusia, menghindari kecelakaan, dan mengatasi dilema etika.

Dalam penelitian AI saat ini, penyelarasan biasanya dicapai dengan mengembangkan tujuan pengganti yang analog dengan tujuan sebenarnya dan mengarahkan sistem AI ke arah tujuan ini. Namun, merancang tujuan penyelarasan AI sangat sulit. Salah satu alasannya adalah bahwa nilai-nilai manusia beragam dan sering kali bergantung pada konteks, sehingga sulit untuk menyaringnya menjadi satu fungsi tujuan yang dapat diterapkan secara universal. Selain itu, kompleksitas lingkungan dunia nyata, di mana nilai dan tujuan sering kali bertentangan atau berkembang seiring waktu, semakin mempersulit upaya penyelarasan. Bahkan jika kita dapat menentukan tujuan yang tepat, sistem AI mungkin menemukan cara yang tidak diinginkan untuk mencapainya, yang mengarah pada hasil yang "tidak selaras" yang secara teknis masih memenuhi tujuan tetapi dengan cara yang berbahaya atau kontraproduktif.

Tantangan-tantangan ini telah memotivasi dan terus memotivasi penelitian AI menuju sistem yang lebih selaras, baik melalui pengembangan mekanisme baru untuk memahami dunia atau metode yang lebih efisien dan dapat digeneralisasi untuk mengadaptasi sistem ini ke tugas-tugas yang diberikan. Yang lebih penting, karena sistem AI menjadi lebih kuat dan cerdas, terutama mengingat bahwa kemajuan terkini dalam LLM telah menunjukkan kemampuan luar biasa dalam menangani banyak masalah yang menantang, kebutuhan akan penyelarasan AI menjadi lebih mendesak. Para peneliti mulai memperhatikan keamanan AI dan memperingatkan masyarakat bahwa mereka perlu mengembangkan dan merilis sistem AI dengan sangat hati-hati untuk mencegah sistem ini tidak selaras [[Russell, 2019; Bengio et al., 2024](#)].

Bibliografi

- [Ainslie dkk., 2020] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, dan Li Yang. Dkk: Pengodean masukan yang panjang dan terstruktur dalam transformer. Dalam Prosiding Konferensi 2020 tentang Metode Empiris dalam Pemrosesan Bahasa Alami (EMNLP), halaman 268–284, 2020.
- [Ainslie dkk., 2023] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, dan Sumit Sanghai. Gqa: Melatih model transformator multikueri umum dari titik pemeriksaan multikepala. Dalam Prosiding Konferensi 2023 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 4895–4901, 2023.
- [Akyürek dkk., 2023] Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, dan Denny Zhou. Algoritma pembelajaran apakah yang dimaksud dengan pembelajaran dalam konteks? investigasi dengan model linear. Dalam Prosiding Konferensi Internasional Kesebelas tentang Representasi Pembelajaran, 2023.
- [Alabdulmohsin dkk., 2022] Ibrahim M Alabdulmohsin, Behnam Neyshabur, dan Xiaohua Zhai. Meninjau kembali hukum penskalaan saraf dalam bahasa dan penglihatan. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 35: 22300–22312, 2022.
- [Allal dkk., 2024] Loubna Ben Allal, Anton Lozhkov, dan Daniel van Strien. kosmopedia: cara membuat data sintetis skala besar untuk pra-pelatihan. <https://huggingface.co/blog/cosmopedia>, Tahun 2024.
- [Almazrouei dkk., 2023] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cap-pelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, dan Guilherme Penedo. Seri elang model bahasa terbuka. arXiv pracetak arXiv:2311.16867, 2023.
- [Andreas dkk., 2016] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, dan Dan Klein. Jaringan modul saraf. Dalam Prosiding konferensi IEEE tentang visi komputer dan pengenalan pola, halaman 39–48, 2016.
- [Arjovsky dkk., 2016] Martin Arjovsky, Amar Shah, dan Yoshua Bengio. Jaringan saraf berulang evolusi kesatuan. Dalam konferensi internasional tentang pembelajaran mesin, halaman 1120–1128, 2016.
- [Aschenbrenner, 2024] Leopold Aschenbrenner. Kesadaran situasional: Dekade ke depan, 2024. URL <https://situational-awareness.ai/>.
- [Askell dkk., 2021] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Benjamin Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, dan Jared Kaplan. Asisten bahasa umum sebagai laboratorium untuk penyelarasan. Pracetak arXiv arXiv:2112.00861, 2021.
- [Bach et al., 2022] Stephen H. Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M. Saiful Bari, Thibault Févry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-David, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Alan Fries, Maged Saeed AlShaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir R. Radev, Mike Tian-Jian Jiang, dan Alexander M. Rush. Promptsouce: Lingkungan pengembangan terpadu dan repositori untuk perintah bahasa alami. Dalam Prosiding Pertemuan Tahunan ke-60 Asosiasi Linguistik Komputasional: Demonstrasi Sistem, halaman 93–104, 2022.
- [Bengio dkk., 2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, dan Christian Jauvin. Model bahasa probabilistik neural. Jurnal Penelitian Pembelajaran Mesin, 3:1137–1155, 2003.
- [Bengio dkk., 2006] Yoshua Bengio, Pascal Lamblin, Dan Popovici, dan Hugo Larochelle. Pelatihan Greedy berlapis-lapis pada jaringan dalam. Kemajuan dalam sistem pemrosesan informasi saraf, 19, 2006.
- [Bengio dkk., 2024] Yoshua Bengio, Geoffrey Hinton, Andrew Yao, Dawn Song, Pieter Abbeel, Trevor Darrell, Yuval Noah Harari, Ya-Qin Zhang, Lan Xue, Shai Shalev-Shwartz, Gillian K. Hadfield, Jeff

Clune, Tegan Maharaj, Frank Hutter, Atilim Gunes Baydin, Sheila A. McIlraith, Qiqi Gao, Ashwin Acharya, David Krueger, Anca Dragan, Philip Torr, Stuart Russell, Daniel Kahneman, Jan Markus Brauner, dan Sören Mindermann. Mengelola risiko AI ekstrem di tengah kemajuan pesat. *Science*, 384(6698): 842–845, 2024.

[Bentivogli dan Giampiccolo, 2011] Luisa Bentivogli dan Danilo Giampiccolo. Pascal mengenali tekstual tantangan implikasi (rte-7) di tac 2011. <https://tac.nist.gov/2011/RTE/>, tahun 2011.

[Besta dkk., 2024] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczek, dan Torsten Hoefer. Grafik pikiran: Memecahkan masalah rumit dengan model bahasa yang besar. Dalam Prosiding Konferensi AAAI tentang Kecerdasan Buatan, volume 38, halaman 17682–17690, 2024.

[Biderman dkk., 2021] Stella Biderman, Sid Black, Charles Foster, Leo Gao, Eric Hallahan, Horace He, Ben Wang, dan Phil Wang. Rotary embeddings: Sebuah revolusi relatif. <https://blog.eleuther.ai/rotary-embeddings/> , Tahun 2021.

[Bishop, 2006] Christopher M. Bishop. Pengenalan Pola dan Pembelajaran Mesin. Springer, 2006.

[Blum dan Mitchell, 1998] Avrim Blum dan Tom Mitchell. Menggabungkan data berlabel dan tidak berlabel dengan pelatihan bersama. Dalam Prosiding konferensi tahunan kesebelas tentang teori pembelajaran komputasional, halaman 92–100, 1998.

[Bradley dan Terry, 1952] Ralph Allan Bradley dan Milton E. Terry. Analisis peringkat blok yang tidak lengkap desain: I. metode perbandingan berpasangan. *Biometrika*, 39(3/4):324–345, 1952.

[Brandon dkk., 2024] William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, dan Jonathan Ragan Kelly. Mengurangi ukuran cache nilai kunci transformer dengan perhatian lintas lapisan. arXiv preprint arXiv:2405.12981, 2024.

[Brill, 1992] Eric Brill. Sebuah penanda bagian ujaran berbasis aturan sederhana. Dalam Ucapan dan Bahasa Alami: Prosiding Lokakarya yang Diadakan di Harriman, New York, 23–26 Februari 1992, 1992.

[Brown dkk., 1993] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, dan Robert L. Mercer. Matematika penerjemahan mesin statistik: Estimasi parameter. *Linguistik Komputasional*, 19(2):263–311, 1993.

[Brown dkk., 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, dan Dario Amodei. Model bahasa adalah pembelajar yang hanya memiliki sedikit kesempatan. Kemajuan dalam sistem pemrosesan informasi saraf, 33:1877–1901, 2020.

[Bubeck et al., 2023] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, dan Yi Zhang. Percikan kecerdasan umum buatan: Eksperimen awal dengan gpt-4. arXiv pracetak arXiv:2303.12712, 2023.

[Bulatov dkk., 2022] Aydar Bulatov, Yury Kuratov, dan Mikhail Burtsev. Transformator memori berulang. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 35:11079–11091, 2022.

[Burges dkk., 2005] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, dan Greg Hullender. Belajar memberi peringkat menggunakan penurunan gradien. Dalam Prosiding konferensi internasional ke-22 tentang Pembelajaran mesin, halaman 89–96, 2005.

[Burns dkk., 2023] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, dan Jeff Wu. Generalisasi lemah ke kuat: Memunculkan kapabilitas kuat dengan supervisi lemah. Pracetak arXiv arXiv:2312.09390, 2023a.

[Burns dkk., 2023] Collin Burns, Jan Leike, Leopold Aschenbrenner, Jeffrey Wu, Pavel Izmailov, Leo Gao, Bowen Baker, dan Jan Hendrik Kirchner. Generalisasi lemah ke kuat, 2023b. URL <https://openai.com/index/generalisasi-lemah-ke-kuat>.

[Caballero dkk., 2023] Ethan Caballero, Kshitij Gupta, Irina Rish, dan David Krueger. Hukum penskalaan saraf yang rusak. Dalam Lokakarya ICLR 2023 tentang Pemahaman Matematis dan Empiris tentang Model Fondasi, 2023.

[Cao dkk., 2007] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, dan Hang Li. Belajar memberi peringkat: dari pendekatan berpasangan ke pendekatan daftar. Dalam Prosiding konferensi internasional ke-24 tentang Pembelajaran mesin, halaman 129–136, 2007.

[Chang et al., 2024] Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Tong Xiao, dan Jingbo Zhu. Metode dorongan yang efisien untuk model bahasa besar: Sebuah survei. arXiv pracetak arXiv:2404.01077, 2024.

[Charniak, 1997] Eugene Charniak. Penguraian statistik dengan tata bahasa bebas konteks dan statistik kata. AAAI/IAAI, 2005(598-603):18, 1997.

[Chen dkk., 2023] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, dan Shengxin Zhu. Melepaskan potensi rekayasa cepat dalam model bahasa besar: tinjauan komprehensif. arXiv preprint arXiv:2310.14735, 2023a.

[Chen et al., 2023] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, dan Hongxia Jin. Alpagasus: Melatih alpaka yang lebih baik dengan data yang lebih sedikit. arXiv pracetak arXiv:2307.08701, 2023b.

[Chen dkk., 2024] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, dan Hongxia Jin. Alpagasus: Melatih alpaka yang lebih baik dengan data yang lebih sedikit. Dalam Konferensi Internasional Keduabelas tentang Representasi Pembelajaran, 2024a.

[Chen dkk., 2023] Shouyuan Chen, Sherman Wong, Liangjian Chen, dan Yuandong Tian. Memperluas jendela konteks model bahasa besar melalui interpolasi posisi. arXiv preprint arXiv:2306.15595, 2023c.

[Chen dkk., 2020] Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, dan Michael Carbin. Hipotesis tiket lotre untuk jaringan bert yang telah dilatih sebelumnya. Kemajuan dalam sistem pemrosesan informasi saraf, 33:15834–15846, 2020.

[Chen dkk., 2024] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, dan Quanquan Gu. Penyempurnaan permainan mandiri mengubah model bahasa yang lemah menjadi model bahasa yang kuat. arXiv pracetak arXiv:2401.01335, 2024b.

[Chevalier dkk., 2023] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, dan Danqi Chen. Mengadaptasi model bahasa untuk mengompresi konteks. Dalam Prosiding Konferensi 2023 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 3829–3846, 2023.

[Chi dkk., 2022] Ta-Chung Chi, Ting-Han Fan, Peter J Ramadge, dan Alexander Rudnicky. Kerple: Penanaman posisi relatif yang dikernelisasi untuk ekstrapolasi panjang. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 35:8386–8399, 2022.

[Chi dkk., 2023] Ta-Chung Chi, Ting-Han Fan, Alexander Rudnicky, dan Peter Ramadge. Membedah ekstrapolasi panjang transformator melalui analisis medan reseptif. Dalam Prosiding Pertemuan Tahunan ke-61 Asosiasi Linguistik Komputasional (Volume 1: Makalah Panjang), halaman 13522–13537, 2023.

[Chiang dkk., 2023] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lian-min Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, dan Eric P. Xing. Vicuna: Chatbot sumber terbuka yang mengesankan gpt-4 dengan kualitas chatgpt 90%*, Maret 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.

[Chowdhery dkk., 2022] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav

Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghe-mawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankara-narayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, dan Noah Fiedel. Palm: Menskalakan pemodelan bahasa dengan jalur. arXiv pracetak arXiv:2204.02311, 2022.

[Christiano dkk., 2017] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, dan Dario Amodei.

Pembelajaran penguatan mendalam dari preferensi manusia. Kemajuan dalam sistem pemrosesan informasi saraf, 30, 2017.

[Chu et al., 2023] Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, dan Ting Liu. Sebuah survei penalaran rantai pemikiran: Kemajuan, batas-batas dan masa depan. arXiv pracetak arXiv:2309.15402, 2023.

[Chung et al., 2022] Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gau-rav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, dan Jason Wei.

Skala model bahasa yang disesuaikan dengan instruksi. Pracetak arXiv arXiv:2210.11416, 2022.

[Clark dkk., 2019] Kevin Clark, Minh-Thang Luong, Quoc V Le, dan Christopher D Manning. Electra: Pra-pelatihan encoder teks sebagai pembeda, bukan generator. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2019.

[Cobbe dkk., 2021] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, dan John Schulman. Melatih verifikator untuk memecahkan soal cerita matematika. Pracetak arXiv arXiv:2110.14168, 2021.

[Conneau et al., 2020] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guil-laume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, dan Veselin Stoyanov. Pembelajaran representasi lintas bahasa tanpa pengawasan dalam skala besar. Dalam Prosiding Pertemuan Tahunan ke-58 Asosiasi Linguistik Komputasional, halaman 8440–8451, 2020.

[Coste dkk., 2024] Thomas Coste, Usman Anwar, Robert Kirk, dan David Krueger. Rangkaian model penghargaan membantu mengurangi pengoptimalan berlebihan. Dalam Konferensi Internasional Kedua Belas tentang Representasi Pembelajaran, 2024.

[Cui dkk., 2024] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, dan Maosong Sun. ULTRAFEEDBACK: Meningkatkan model bahasa dengan umpan balik AI berskala. Dalam Proceedings of the 41st International Conference on Machine Learning, volume 235, halaman 9722–9744, 2024.

[Dai et al., 2023] Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, dan Furu Wei.

Mengapa gpt dapat belajar dalam konteks? model bahasa secara diam-diam melakukan penurunan gradien sebagai meta-pengoptimal. Dalam Temuan Asosiasi Linguistik Komputasi: ACL 2023, halaman 4005–4019, 2023.

[Dai dkk., 2019] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, dan Ruslan Salakhutdinov. Transformer-xl: Model bahasa yang penuh perhatian di luar konteks dengan durasi tetap. Dalam Prosiding Pertemuan Tahunan ke-57 Asosiasi Linguistik Komputasional, halaman 2978–2988, 2019.

[Dao et al., 2022] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, dan Christopher Ré. Flashattention: Perhatian tepat yang cepat dan hemat memori dengan kesadaran io. Kemajuan dalam Pemrosesan Informasi Neural

Sistem, 35:16344–16359, 2022.

[Dehghani dkk., 2018] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, dan ýukasz Kaiser. Transformator universal. Pracetak arXiv arXiv:1807.03819, 2018.

[Deletang dkk., 2024] Gregoire Deletang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, dan Joel Veness. Pemodelan bahasa adalah kompresi. Dalam Konferensi Internasional Kedua Belas tentang Representasi Pembelajaran, 2024.

[Deng dkk., 2022] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, dan Zhitong Hu. Rlprompt: Mengoptimalkan perintah teks diskrit dengan pembelajaran penguatan. Dalam Prosiding Konferensi 2022 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 3369–3391, 2022.

[Devlin dkk., 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, dan Kristina Toutanova. Bert: Pra-pelatihan transformator dwiarah dalam untuk pemahaman bahasa. Dalam Prosiding Konferensi 2019 North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Makalah Panjang dan Pendek), halaman 4171–4186, 2019.

[Ding et al., 2024] Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, dan Mao Yang. Longrope: Memperluas jendela konteks II melampaui 2 juta token. arXiv pracetak arXiv:2402.13753, 2024.

[Dolan dan Brockett, 2005] Bill Dolan dan Chris Brockett. Secara otomatis menyusun korpus parafrase kalimat. Dalam Prosiding Lokakarya Internasional Ketiga tentang Parafrase (IWP2005), 2005.

[Dong et al., 2019] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, dan Hsiao-Wuen Hon. Pra-pelatihan model bahasa terpadu untuk pemahaman dan pembangkitan bahasa alami. Kemajuan dalam sistem pemrosesan informasi saraf, 32, 2019.

[Dong dkk., 2022] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, dan Zhifang Sui. Sebuah survei tentang pembelajaran dalam konteks. arXiv pracetak arXiv:2301.00234, 2022.

[Dong et al., 2021] Yihe Dong, Jean-Baptiste Cordonnier, dan Andreas Loukas. Perhatian bukanlah satu-satunya yang Anda butuhkan: Perhatian murni kehilangan peringkat dua kali lipat secara eksponensial seiring dengan kedalaman. Dalam Konferensi Internasional tentang Pembelajaran Mesin, halaman 2793–2803. PMLR, 2021.

[Drozdov dkk., 2022] Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, dan Denny Zhou. Penguraian semantik komposisi dengan model bahasa besar. Dalam Prosiding Konferensi Internasional Kesebelas tentang Representasi Pembelajaran, 2022.

[Dua dkk., 2022] Dheeru Dua, Shivanshu Gupta, Sameer Singh, dan Matt Gardner. Dorongan berurutan untuk menguraikan pertanyaan kompleks. Dalam Prosiding Konferensi 2022 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 1251–1265, 2022.

[Dubey dkk., 2024] Abimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, dkk. Kawan model llama 3. arXiv pracetak arXiv:2407.21783, 2024.

[Dubois dkk., 2024] Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, dan Tatsunori B Hashimoto. Alpacafarm: Kerangka simulasi untuk metode yang belajar dari umpan balik manusia. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 36, 2024.

[Eisenstein dkk., 2023] Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, dan Peter Shaw. Membantu atau menggembalakan? Kumpulan model penghargaan mengurangi tetapi tidak menghilangkan peretasan penghargaan. Pracetak arXiv arXiv:2312.09244, 2023.

[Elsken dkk., 2019] Thomas Elsken, Jan Hendrik Metzen, dan Frank Hutter. Pencarian arsitektur saraf: Sebuah survei. Jurnal Penelitian Pembelajaran Mesin, 20(55):1–21, 2019.

- [Erhan dkk., 2010] Dumitru Erhan, Aaron Courville, Yoshua Bengio, dan Pascal Vincent. Mengapa pelatihan awal tanpa pengawasan membantu pembelajaran mendalam? Dalam Prosiding konferensi internasional ketiga belas tentang kecerdasan buatan dan statistik, halaman 201–208, 2010.
- [Fan dkk., 2019] Angela Fan, Edouard Grave, dan Armand Joulin. Mengurangi kedalaman transformator sesuai permintaan dengan pemutusan terstruktur. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2019.
- [Fedus et al., 2022] William Fedus, Barret Zoph, dan Noam Shazeer. Transformator sakelar: Penskalaan ke model parameter triliun dengan kelangkaan sederhana dan efisien. Jurnal Penelitian Pembelajaran Mesin, 23(1):5232–5270, 2022.
- [Fernandes dkk., 2023] Patrick Fernandes, Aman Madaan, Emmy Liu, António Farinhos, Pedro Henrique Martins, Amanda Bertsch, José GC de Souza, Shuyan Zhou, Tongshuang Wu, Graham Neubig, dan André FT Martins. Menjembatani kesenjangan: Sebuah survei tentang pengintegrasian umpan balik (manusia) untuk generasi bahasa alami. Transaksi Asosiasi Linguistik Komputasi, 11:1643–1668, 2023.
- [Franklin dan Graesser, 1996] Stan Franklin dan Art Graesser. Apakah itu agen, atau hanya program?: Taksonomi untuk agen otonom. Dalam lokakarya internasional tentang teori agen, arsitektur, dan bahasa, halaman 21–35. Springer, 1996.
- [Frensch dan Funke, 2014] Peter A Frensch dan Joachim Funke. Pemecahan masalah kompleks: Pendekatan Eropa perspektif. Psychology Press, 2014.
- [Gale et al., 2019] Trevor Gale, Erich Elsen, dan Sara Hooker. Keadaan kelangkaan dalam jaringan saraf dalam. arXiv pracetak arXiv:1902.09574, 2019.
- [Ganguli dkk., 2023] Deep Ganguli, Amanda Askell, Nicholas Schiefer, Thomas I. Liao, Kamile Luko-siute, Anna Chen, Anna Goldie, Azalia Mirhoseini, Catherine Olsson, Danny Hernandez, Dawn Drain, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jackson Kernion, Jamie Kerr, Jared Mueller, Joshua Landau, Kamal Ndousse, Karina Nguyen, Liane Lovitt, Michael Sellitto, Nelson Elhage, Noemí Mercado, Nova DasSarma, Oliver Rausch, Robert Lasenby, Robin Larson, Sam Ringer, Sandipan Kundu, Saurav Kada-vath, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, Christopher Olah, Jack Clark, Samuel R. Bowman, dan Jared Kaplan. Kapasitas untuk koreksi moral diri dalam model bahasa yang besar. arXiv preprint arXiv:2302.07459, 2023.
- [Gao dkk., 2023] Leo Gao, John Schulman, dan Jacob Hilton. Hukum penskalaan untuk pengoptimalan model imbalan yang berlebihan. Dalam Konferensi Internasional tentang Pembelajaran Mesin, halaman 10835–10866. PMLR, 2023a.
- [Gao dkk., 2023] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, dan Graham Neubig. Sobat: Model bahasa berbantuan program. Dalam Konferensi Internasional tentang Pembelajaran Mesin, halaman 10764–10799. PMLR, 2023b.
- [Gao dkk., 2023] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, dan Haofen Wang. Generasi yang ditambah pengambilan untuk model bahasa besar: Sebuah survei. arXiv pracetak arXiv:2312.10997, 2023c.
- [Garg dkk., 2022] Shivam Garg, Dimitris Tsipras, Percy S Liang, dan Gregory Valiant. Apa yang dapat dipelajari para transformator dalam konteks studi kasus kelas fungsi sederhana. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 35:30583–30598, 2022.
- [Ge et al., 2024] Yuan Ge, Yilun Liu, Chi Hu, Weibin Meng, Shimin Tao, Xiaofeng Zhao, Hongxia Ma, Li Zhang, Boxing Chen, Hao Yang, Bei Li, Tong Xiao, dan Jingbo Zhu. Pengelompokan dan pemeringkatan: Pemilihan instruksi yang menjaga keberagaman melalui estimasi kualitas yang selaras dengan pakar. arXiv pracetak arXiv:2402.18191, 2024.
- [Tim Gemma, 2024] Tim Gemma Google DeepMind. Gemma: Model Terbuka Berdasarkan Gemini Re-pencarian dan Teknologi, 2024.
- [Goodhart, 1984] Charles AE Goodhart. Masalah manajemen moneter: pengalaman Inggris. Peloncat,

- [Gordon dkk., 2021] Mitchell A Gordon, Kevin Duh, dan Jared Kaplan. Hukum penskalaan data dan parameter untuk penerjemahan mesin neural. Dalam Prosiding Konferensi 2021 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 5915–5922, 2021.
- [Gu dan Dao, 2023] Albert Gu dan Tri Dao. Mamba: Pemodelan deret waktu linier dengan ruang keadaan selektif. arXiv preprint arXiv:2312.00752, 2023.
- [Gunasekar dkk., 2023] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripour, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, dan Yuanzhi Li. Buku teks adalah semua yang Anda butuhkan. arXiv pracetak arXiv:2306.11644, 2023.
- [Guo dkk., 2024] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, dan Yujiu Yang. Menghubungkan model bahasa yang besar dengan algoritma evolusi menghasilkan pengoptimal cepat yang hebat. Dalam Konferensi Internasional Kedua Belas tentang Representasi Pembelajaran, 2024.
- [Gupta dan Berant, 2020] Ankit Gupta dan Jonathan Berant. Gmat: Augmentasi memori global untuk transformator. pracetak arXiv arXiv:2006.03274, 2020.
- [Gupta dkk., 2021] Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, dan Jonathan Berant. Transformator hemat memori melalui perhatian top-k. Dalam Prosiding Lokakarya Kedua tentang Pemrosesan Bahasa Alami yang Sederhana dan Efisien, halaman 39–52, 2021.
- [Han et al., 2021] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezong Qiu, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, dan Jun Zhu. Model terlatih: Dulu, sekarang, dan masa depan. AI Terbuka, 2:225–250, 2021.
- [Han dkk., 2024] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, dan Sai Qian Zhang. Penyetelan halus yang efisien terhadap parameter untuk model besar: Survei komprehensif. arXiv preprint arXiv:2403.14608, 2024.
- [Harlap dkk., 2018] Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil Deva-nur, Greg Ganger, dan Phil Gibbons. Pipedream: Pelatihan dnn paralel pipeline yang cepat dan efisien. arXiv pracetak arXiv:1806.03377, 2018.
- [He et al., 2019] Kaiming He, Ross Girshick, dan Piotr Dollár. Memikirkan kembali pra-pelatihan imagenet. Dalam Prosiding Konferensi Internasional IEEE/CVF tentang Visi Komputer, halaman 4918–4927, 2019.
- [He et al., 2021] Pengcheng He, Xiaodong Liu, Jianfeng Gao, dan Weizhu Chen. Deberta: Peningkatan decoding bert dengan perhatian yang tidak terikat. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2021.
- [Hendrycks dan Gimpel, 2016] Dan Hendrycks dan Kevin Gimpel. Satuan linier galat Gaussian (gelus). arXiv pracetak arXiv:1606.08415, 2016.
- [Hendrycks dkk., 2020] Dan Hendrycks, Xiaoyuan Liu, Eric Wallace, Adam Dziedzic, Rishabh Krishnan, dan Dawn Song. Transformator yang telah dilatih sebelumnya meningkatkan ketahanan di luar distribusi. Dalam Prosiding Pertemuan Tahunan ke-58 Asosiasi Linguistik Komputasional, halaman 2744–2751, 2020.
- [Hendrycks dkk., 2021] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, dan Jacob Steinhardt. Mengukur pemahaman bahasa dalam berbagai tugas. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2021.
- [Hestness dkk., 2017] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, dan Yanqi Zhou. Penskalaan pembelajaran mendalam dapat diprediksi secara empiris. arXiv pracetak arXiv:1712.00409, 2017.
- [Hewitt, 2024] John Hewitt. Mengikuti instruksi tanpa menyetel instruksi, 2024. URL <https://nlp.stanford.edu/~johnhew/instruction-following.html>.
- [Hewitt dkk., 2024] John Hewitt, Nelson F Liu, Percy Liang, dan Christopher D Manning. Mengikuti instruksi tanpa menyetel instruksi. arXiv preprint arXiv:2409.14254, 2024.

- [Hochreiter dan Schmidhuber, 1997] Sepp Hochreiter dan Jürgen Schmidhuber. Memori jangka pendek. *Komputasi saraf*, 9(8):1735–1780, 1997.
- [Hoffmann dkk., 2022] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Si-mon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, dan Laurent Sifre. Melatih model bahasa besar yang optimal untuk komputasi. Pracetak arXiv arXiv:2203.15556, 2022.
- [Honovich dkk., 2023] Atau Honovich, Thomas Scialom, Omer Levy, dan Timo Schick. Instruksi yang tidak alami: Menyetel model bahasa tanpa (hampir) tenaga manusia. Dalam Prosiding Pertemuan Tahunan ke-61 Asosiasi Linguistik Komputasional (Volume 1: Makalah Panjang), halaman 14409–14428, 2023.
- [Houlsby dkk., 2019] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, dan Sylvain Gelly. Pembelajaran transfer yang efisien parameter untuk NLP. Dalam Prosiding Konferensi Internasional ke-36 tentang Pembelajaran Mesin, halaman 2790–2799. PMLR, 2019.
- [Hu et al., 2022] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, dan Weizhu Chen. LORA: Adaptasi tingkat rendah dari model bahasa besar. Dalam Konferensi Internasional tentang Representasi Pembelajaran, 2022.
- [Huang, 2009] Liang Huang. Algoritma pencarian berbasis pemrograman dinamis dalam NLP. Dalam Prosiding Teknologi Bahasa Manusia: Konferensi Tahunan 2009 dari Asosiasi Linguistik Komputasional Cabang Amerika Utara, Volume Pendamping: Abstrak Tutorial, 2009.
- [Huang et al., 2019] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, Hyouk-Joong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, dan Zhifeng Chen. Gpipe: Pelatihan jaringan saraf raksasa yang efisien menggunakan paralelisme pipa. Kemajuan dalam sistem pemrosesan informasi saraf, 32, 2019.
- [Hutchins dkk., 2022] DeLesley Hutchins, Imanol Schlag, Yuhuai Wu, Ethan Dyer, dan Behnam Neyshabur. Transformator blok-rekursif. Kemajuan dalam sistem pemrosesan informasi saraf, 35: 33248–33261, 2022.
- [Jelinek, 1998] Frederick Jelinek. Metode statistik untuk pengenalan suara. MIT Press, 1998.
- [Jiang dkk., 2023] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Deven-dra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, dan William El Sayed. Mistral 7b. arXiv pracetak arXiv:2310.06825, 2023a.
- [Jiang dkk., 2023] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, dan Lili Qiu. LimLingua: Kompresi perintah untuk mempercepat inferensi model bahasa besar. Dalam Prosiding Konferensi 2023 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 13358–13376, 2023b.
- [Jiang dkk., 2020] Zhengbao Jiang, Frank F Xu, Jun Araki, dan Graham Neubig. Bagaimana kita dapat mengetahui apa yang diketahui oleh model bahasa? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.
- [Jiao dkk., 2020] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, dan Qun Liu. Tinybert: Menyinkirkkan bert untuk pemahaman bahasa alami. Dalam Temuan Asosiasi Linguistik Komputasi: EMNLP 2020, halaman 4163–4174, 2020.
- [Joshi dkk., 2017] Mandar Joshi, Eunsol Choi, Daniel S Weld, dan Luke Zettlemoyer. Triviaqa: Kumpulan data tantangan pengawasan jarak jauh berskala besar untuk pemahaman bacaan. Dalam Prosiding Pertemuan Tahunan ke-55 Asosiasi Linguistik Komputasional (Volume 1: Makalah Panjang), halaman 1601–1611, 2017.
- [Joshi et al., 2020] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, dan Omer Levy. Spanbert: Meningkatkan pra-pelatihan dengan merepresentasikan dan memprediksi rentang. Transaksi

- asosiasi untuk linguistik komputasional, 8:64–77, 2020.
- [Jurafsky dan Martin, 2008] Dan Jurafsky dan James H. Martin. Pemrosesan Ucapan dan Bahasa (edisi ke-2) (ed.). Prentice Hall, 2008.
- [Kahneman, 2011] Daniel Kahneman. Berpikir, cepat dan lambat. Macmillan, 2011.
- [Kaplan dkk., 2020] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Re-won Child, Scott Gray, Alec Radford, Jeffrey Wu, dan Dario Amodei. Hukum penskalaan untuk model bahasa neural. arXiv preprint arXiv:2001.08361, 2020.
- [Katharopoulos dkk., 2020] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, dan François Fleuret. Transformer adalah rnns: Transformer autoregresif cepat dengan perhatian linier. Dalam konferensi internasional tentang pembelajaran mesin, halaman 5156–5165. PMLR, 2020.
- [Khandelwal dkk., 2020] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, dan Mike Lewis. Generalisasi melalui hafalan: Model bahasa tetangga terdekat. Dalam Konferensi Internasional tentang Representasi Pembelajaran, 2020.
- [Khot dkk., 2023] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, dan Ashish Sabharwal. Decomposed prompting: Pendekatan modular untuk menyelesaikan tugas-tugas kompleks. Dalam Prosiding Konferensi Internasional Kesebelas tentang Representasi Pembelajaran, 2023.
- [Kim dkk., 2023] Sehoon Kim, Coleman Hooper, Thanakul Wattanawong, Minwoo Kang, Ruohan Yan, Hasan Genc, Grace Dinh, Qijing Huang, Kurt Keutzer, Michael W. Mahoney, Yakun Sophia Shao, dan Amir Gholami. Optimasi tumpukan penuh inferensi transformator: survei. arXiv preprint arXiv:2302.14017, 2023.
- [Kirkpatrick dkk., 2017] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharsan Kumaran, dan Raia Hadsell. Mengatasi bencana lupa di jaringan saraf. Prosiding akademi ilmu pengetahuan nasional, 114(13):3521–3526, 2017.
- [Koehn, 2010] Philipp Koehn. Terjemahan Mesin Statistik. Cambridge University Press, 2010.
- [Kojima dkk., 2022] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, dan Yusuke Iwasawa. Model bahasa besar adalah penalaran zero-shot. Kemajuan dalam sistem pemrosesan informasi saraf, 35:22199–22213, 2022.
- [Korthikanti dkk., 2023] Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, dan Bryan Catanzaro. Mengurangi penghitungan ulang aktivasi dalam model transformator besar. Prosiding Machine Learning and Systems, 5, 2023.
- [Krakovna dkk., 2020] Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, dan Shane Legg. Spesifikasi permainan: sisi lain dari kecerdikan AI. <https://deepmind.google/discover/blog/specifi-gaming-the-flip-side-of-ai-ingenuity>, Tahun 2020.
- [Kung dan Peng, 2023] Po-Nien Kung dan Nanyun Peng. Apakah model benar-benar belajar mengikuti instruksi? studi empiris tentang penyetelan instruksi. arXiv pracetak arXiv:2305.11383, 2023.
- [Kwon dkk., 2023] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E Gonzalez, Hao Zhang, dan Ion Stoica. Manajemen memori yang efisien untuk penyajian model bahasa besar dengan pagedattention. arXiv preprint arXiv:2309.06180, 2023.
- [Lake dan Baroni, 2018] Brenden Lake dan Marco Baroni. Generalisasi tanpa sistematisitas: Mengenai keterampilan komposisi jaringan berulang urutan-ke-urutan. Dalam konferensi internasional tentang pembelajaran mesin, halaman 2873–2882. PMLR, 2018.
- [Lambert dkk., 2024] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, dan Han-naneh Hajishirzi. Rewardbench: Mengevaluasi model penghargaan untuk pemodelan bahasa. Pracetak arXiv

arXiv:2403.13787, 2024.

[Lample dan Conneau, 2019] Guillaume Lample dan Alexis Conneau. Model bahasa lintas bahasa prapelatihan. *pracetak arXiv arXiv:1901.07291*, 2019.

[Lan dkk., 2020] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, dan Radu Soricut. Albert: Buku pegangan untuk pembelajaran representasi bahasa yang diawasi sendiri. Dalam Prosiding Konferensi Internasional tentang Pembelajaran Representasi, 2020.

[Lee dkk., 2023] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Ren Lu, Thomas Mesnard, Johan Ferret, Colton Bishop, Ethan Hall, Victor Carbune, dan Abhinav Rastogi. Rlaif: Meningkatkan pembelajaran penguatan dari umpan balik manusia dengan umpan balik AI. *arXiv preprint arXiv:2309.00267*, 2023.

[Lester dkk., 2021] Brian Lester, Rami Al-Rfou, dan Noah Constant. Kekuatan skala untuk penyetelan cepat yang efisien parameter. Dalam Prosiding Konferensi 2021 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 3045–3059, 2021.

[Lewis dkk., 2020] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mo-hamed, Omer Levy, Veselin Stoyanov, dan Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training untuk pembangkitan, penerjemahan, dan pemahaman bahasa alami. Dalam Prosiding Pertemuan Tahunan ke-58 Asosiasi Linguistik Komputasional, halaman 7871–7880, 2020.

[Li et al., 2023] Bei Li, Rui Wang, Junliang Guo, Kaitao Song, Xu Tan, Hany Hassan, Arul Menezes, Tong Xiao, Jiang Bian, dan JingBo Zhu. Deliberate then generate: Kerangka kerja prompting yang disempurnakan untuk pembuatan teks. *arXiv preprint arXiv:2305.19835*, 2023a.

[Li, 2011] Hang Li. Belajar Memberi Peringkat untuk Pengambilan Informasi dan Pemrosesan Bahasa Alami. Akses daring: Morgan & Claypool Synthesis Collection Five. Morgan & Claypool Publishers, 2011. ISBN 9781608457076.

[Li dkk., 2022] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, dan Lemao Liu. Sebuah survei tentang pembuatan teks augmented pengambilan. *arXiv pracetak arXiv:2202.01110*, 2022.

[Li et al., 2024] Shanda Li, Chong You, Guru Guruganesh, Joshua Ainslie, Santiago Ontanon, Manzil Zaheer, Sumit Sanghai, Yiming Yang, Sanjiv Kumar, dan Srinadh Bhojanapalli. Interpolasi fungsional untuk posisi relatif meningkatkan transformator konteks panjang. Dalam Konferensi Internasional Kedua Belas tentang Representasi Pembelajaran, 2024.

[Li et al., 2023] Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, dan Yang You. Paralelisme sekuens: Pelatihan sekuens panjang dari perspektif sistem. Dalam Prosiding Pertemuan Tahunan ke-61 Asosiasi Linguistik Komputasional (Volume 1: Makalah Panjang), halaman 2391–2404, 2023b.

[Li dan Liang, 2021] Xiang Lisa Li dan Percy Liang. Prefix-tuning: Mengoptimalkan perintah berkelanjutan untuk pembuatan. Dalam Prosiding Pertemuan Tahunan ke-59 Asosiasi Linguistik Komputasional dan Konferensi Gabungan Internasional ke-11 tentang Pemrosesan Bahasa Alami (Volume 1: Makalah Panjang), halaman 4582–4597, 2021.

[Li, 2023] Yinheng Li. Survei praktis tentang desain prompt zero-shot untuk pembelajaran dalam konteks. Dalam Prosiding Konferensi Internasional ke-14 tentang Kemajuan Terbaru dalam Pemrosesan Bahasa Alami, halaman 641–647, 2023.

[Li et al., 2023] Yucheng Li, Bo Dong, Frank Guerin, dan Chenghua Lin. Kompresi konteks untuk meningkatkan efisiensi inferensi model bahasa besar. Dalam Prosiding Konferensi 2023 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 6342–6353, 2023c.

[Lialin et al., 2023] Vladislav Lialin, Vijeta Deshpande, dan Anna Rumshisky. Dari skala bawah ke skala atas: Panduan untuk fine-tuning parameter-efisien. *arXiv preprint arXiv:2303.15647*, 2023.

[Lightman dkk., 2024] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, dan Karl Cobbe. Mari kita verifikasi langkah demi langkah. Dalam Konferensi Internasional Kedua Belas tentang Representasi Pembelajaran, 2024.

[Liu dkk., 2024] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang

- Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, dkk. Laporan teknis Deepseek-v3. arXiv pracetak arXiv:2412.19437, 2024a.
- [Liu dkk., 2022] Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, dan Weizhu Chen. Apa yang membuat contoh kontekstual yang baik untuk gpt-3? Dalam Prosiding Deep Learning Inside Out (DeeLIO 2022): Lokakarya ke-3 tentang Ekstraksi dan Integrasi Pengetahuan untuk Arsitektur Deep Learning, halaman 100–114, 2022.
- [Liu dkk., 2023] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, dan Graham Neubig. Pra-pelatihan, perintah, dan prediksi: Survei sistematis metode perintah dalam pemrosesan bahasa alami. Survei Komputasi ACM, 55(9):1–35, 2023a.
- [Liu dkk., 2024] Tianqi Liu, Yao Zhao, Rishabh Joshi, Misha Khalman, Mohammad Saleh, Peter J Liu, dan Jialu Liu. Pengambilan sampel penolakan statistik meningkatkan pengoptimalan preferensi. Dalam Konferensi Internasional Kedua Belas tentang Representasi Pembelajaran, 2024b.
- [Liu, 2009] Tie-Yan Liu. Belajar memberi peringkat untuk pengambilan informasi. Fondasi dan Tren® dalam Informasi Pengambilan kembali, 3(3):225–331, 2009.
- [Liu et al., 2023] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yuje Qian, Zhilin Yang, dan Jie Tang. Gpt paham juga. AI Terbuka, 2023b.
- [Liu et al., 2023] Xiaoxia Liu, Jingyi Wang, Jun Sun, Xiaohan Yuan, Guoliang Dong, Peng Di, Wenhui Wang, dan Dongxia Wang. Kerangka kerja yang mendorong untuk model bahasa besar: Sebuah survei. arXiv pracetak arXiv:2311.12785, 2023c.
- [Liu dkk., 2024] Xinyu Liu, Runsong Zhao, Pengcheng Huang, Chunyang Xiao, Bei Li, Jingang Wang, Tong Xiao, dan Jingbo Zhu. Kurva lupa: Metode yang andal untuk mengevaluasi kemampuan menghafal untuk model konteks panjang. Dalam Prosiding Konferensi 2024 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 4667–4682, 2024c.
- [Liu dkk., 2019] Yinhai Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, dan Veselin Stoyanov. Roberta: Pendekatan prapelatihan bert yang dioptimalkan secara tangguh. arXiv preprint arXiv:1907.11692, 2019.
- [Longpre dkk., 2023] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, dan Adam Roberts. Koleksi flan: Merancang data dan metode untuk penyetelan instruksi yang efektif. Dalam Konferensi Internasional tentang Pembelajaran Mesin, halaman 22631–22648. PMLR, 2023.
- [Ma et al., 2023] Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, dan Luke Zettlemoyer. Mega: Rata-rata bergerak yang dilengkapi perhatian terjaga. Dalam Konferensi Internasional Kesebelas tentang Representasi Pembelajaran, 2023.
- [Ma et al., 2024] Xuezhe Ma, Xiaomeng Yang, Wenhan Xiong, Beidi Chen, Lili Yu, Hao Zhang, Jonathan May, Luke Zettlemoyer, Omer Levy, dan Chunting Zhou. Megalodon: Pra-pelatihan dan inferensi llm yang efisien dengan panjang konteks tak terbatas. Pracetak arXiv arXiv:2404.08801, 2024.
- [Madaan et al., 2024] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, dan Peter Clark. Penyempurnaan mandiri: Penyempurnaan berulang dengan umpan balik mandiri. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 36, 2024.
- [Manning, 2022] Christopher D Manning. Pemahaman dan penalaran bahasa manusia. Daedalus, 151(2):127–138, 2022.
- [Marcus, 1993] Gary F Marcus. Bukti negatif dalam pemerolehan bahasa. Kognisi, 46(1):53–85, 1993.
- [Martins dkk., 2022] Pedro Henrique Martins, Zita Marinho, dan André FT Martins. $\ddot{\gamma}$ -former: Transformator memori tak terbatas-former: Transformator memori tak terbatas. Dalam Prosiding Pertemuan Tahunan ke-60 Asosiasi Linguistik Komputasional (Volume 1: Makalah Panjang), halaman 5468–5485, 2022.

- [Mavi et al., 2024] Vaibhav Mavi, Anubhav Jangra, dan Adam Jatowt. Menjawab pertanyaan multi-hop. *Yayasan dan Tren® dalam Pengambilan Informasi*, 17(5):457–586, 2024.
- [Michel dkk., 2019] Paul Michel, Omer Levy, dan Graham Neubig. Apakah enam belas kepala benar-benar lebih baik daripada satu? *Kemajuan dalam sistem pemrosesan informasi saraf*, 32, 2019.
- [Micikevicius dkk., 2018] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, dan Hao Wu. Pelatihan presisi campuran. Dalam *Prosiding Konferensi Internasional tentang Representasi Pembelajaran*, 2018.
- [Miettinen, 1999] Kaisa Miettinen. Optimasi multiobjektif nonlinier, volume 12. Springer Science & Business Media, 1999.
- [Mikolov dkk., 2013] Tomas Mikolov, Kai Chen, Greg Corrado, dan Jeffrey Dean. Estimasi efisien representasi kata dalam ruang vektor. Dalam *Prosiding Konferensi Internasional tentang Representasi Pembelajaran (ICLR 2013)*, 2013a.
- [Mikolov dkk., 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, dan Jeffrey Dean. Representasi kata dan frasa yang terdistribusi dan komposisinya. Dalam *Prosiding Konferensi Internasional ke-26 tentang Sistem Pemrosesan Informasi Neural - Volume 2*, halaman 3111–3119, 2013b.
- [Min dkk., 2019] Sewon Min, Victor Zhong, Luke Zettlemoyer, dan Hannaneh Hajishirzi. Pemahaman bacaan multi-hop melalui dekomposisi pertanyaan dan penilaian ulang. Dalam *Prosiding Pertemuan Tahunan ke-57 Asosiasi Linguistik Komputasional*, halaman 6097–6109, 2019.
- [Minaee et al., 2024] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, dan Jianfeng Gao. Model bahasa besar: Sebuah survei. *arXiv* pracetak arXiv:2402.06196, 2024.
- [Mishra dkk., 2022] Swaroop Mishra, Daniel Khashabi, Chitta Baral, dan Hannaneh Hajishirzi. Generalisasi lintas tugas melalui instruksi crowdsourcing bahasa alami. Dalam *Prosiding Pertemuan Tahunan ke-60 Asosiasi Linguistik Komputasional (Volume 1: Makalah Panjang)*, halaman 3470–3487, 2022.
- [Mnih dkk., 2016] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P Lillicrap, David Silver, dan Koray Kavukcuoglu. Metode asinkron untuk pembelajaran penguatan mendalam. Dalam *Prosiding Konferensi Internasional ke-33 tentang Konferensi Internasional tentang Pembelajaran Mesin*, halaman 1928–1937, 2016.
- [Mohtashami dan Jaggi, 2024] Amirkeivan Mohtashami dan Martin Jaggi. Konteks tak terbatas akses acak panjang untuk transformator. *Kemajuan dalam Sistem Pemrosesan Informasi Neural*, 36, 2024.
- [Mu et al., 2024] Jesse Mu, Xiang Li, dan Noah Goodman. Belajar mengompresi prompt dengan token gist. *Kemajuan dalam Sistem Pemrosesan Informasi Neural*, 36, 2024.
- [Munkhdalai dkk., 2024] Tsendsuren Munkhdalai, Manaal Faruqui, dan Siddharth Gopal. Jangan tinggalkan konteks: Transformator konteks tak terhingga yang efisien dengan perhatian tak terhingga. *Pracetak arXiv arXiv:2404.07143*, 2024.
- [Nakano dkk., 2021] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, dan John Schulman. Webgpt: Tanya jawab dibantu browser dengan umpan balik manusia. *Pracetak arXiv arXiv:2112.09332*, 2021.
- [Narayanan dkk., 2021] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, dan Matei Zaharia. Pelatihan model bahasa skala besar yang efisien pada cluster GPU menggunakan megatron-lm. Dalam *Prosiding Konferensi Internasional untuk Komputasi, Jaringan, Penyimpanan dan Analisis Kinerja Tinggi*, halaman 1–15, 2021.

- [Ng dkk., 1999] Andrew Y Ng, Daishi Harada, dan Stuart J Russell. Invariansi kebijakan dalam hal imbalan transformasi: Teori dan aplikasi untuk pembentukan penghargaan. Dalam Prosiding Konferensi Internasional Keenambelas Konferensi tentang Pembelajaran Mesin, halaman 278–287, 1999.
- [OpenAI, 2024] OpenAI. Belajar bernalar dengan llms, September 2024. URL <https://openai.com/index/belajar-bernalar-dengan-llms/>.
- [Ouyang dkk., 2022] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, dan Ryan Lowe. Melatih model bahasa untuk mengikuti instruksi dengan umpan balik manusia. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 35:27730–27744, 2022.
- [Pal dkk., 2023] Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C Wallace, dan David Bau. Lensa masa depan: Mengantisipasi token berikutnya dari satu status tersembunyi. Dalam Prosiding Konferensi ke-27 tentang Pembelajaran Bahasa Alami Komputasional (CoNLL), halaman 548–560, 2023.
- [Pan et al., 2022] Alexander Pan, Kush Bhatia, dan Jacob Steinhardt. Dampak kesalahan spesifikasi penghargaan: Pemetaan dan mitigasi model yang tidak selaras. Dalam Konferensi Internasional tentang Representasi Pembelajaran, 2022.
- [Pan et al., 2024] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, dan William Yang Wang. Memperbaiki model bahasa besar secara otomatis: Meninjau lanskap berbagai strategi koreksi otomatis. Transaksi Asosiasi Linguistik Komputasional, 12:484–506, 2024.
- [Parisi dkk., 2022] Aaron Parisi, Yao Zhao, dan Noah Fiedel. Talm: Alat model bahasa yang ditambah. arXiv pracetak arXiv:2205.12255, 2022.
- [Parisi dkk., 2019] Jerman I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, dan Stefan Wermter. Pembelajaran berkelanjutan seumur hidup dengan jaringan saraf: Tinjauan. Jaringan saraf, 113:54–71, 2019.
- [Parmar dkk., 2018] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, dan Dustin Tran. Transformator gambar. Dalam konferensi internasional tentang pembelajaran mesin, halaman 4055–4064. PMLR, 2018.
- [Penedo dkk., 2023] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobaidli, Baptiste Pannier, Ebtesam Almazrouei, dan Julien Launay. Kumpulan data web yang disempurnakan untuk falcon llm: mengungguli corpora yang dikurasi dengan data web, dan data web saja. arXiv pracetak arXiv:2306.01116, 2023.
- [Peng et al., 2024] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, dan Enrico Shippele. YaRN: Ekstensi jendela kontekstual yang efisien dari model bahasa yang besar. Dalam Konferensi Internasional Kedua Belas tentang Pembelajaran Representasi, 2024.
- [Pennington et al., 2014] Jeffrey Pennington, Richard Socher, dan Christopher D. Manning. Glove: Global vektor untuk representasi kata. Dalam Prosiding Metode Empiris dalam Pemrosesan Bahasa Alami (EMNLP), halaman 1532–1543, 2014.
- [Peters dkk., 2018] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, dan Luke Zettlemoyer. Representasi kata kontekstual yang mendalam. Dalam Prosiding Konferensi Cabang Amerika Utara dari Asosiasi Linguistik Komputasi 2018: Teknologi Bahasa Manusia, Volume 1 (Makalah Panjang), 2018.
- [Plackett, 1975] Robin L Plackett. Analisis permutasi. Jurnal Royal Statistics Society Seri C: Statistik Terapan, 24(2):193–202, 1975.
- [Prasad et al., 2023] Archiki Prasad, Peter Hase, Xiang Zhou, dan Mohit Bansal. Grips: Pencarian instruksi berbasis edit tanpa gradien untuk mendorong model bahasa besar. Dalam Prosiding Konferensi ke-17 Bab Eropa dari Asosiasi Linguistik Komputasi, halaman 3845–3864, 2023.

- [Press et al., 2022] Ofir Press, Noah Smith, dan Mike Lewis. Latihan singkat, ujian panjang: Perhatian dengan bias linier memungkinkan ekstrapolasi panjang masukan. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2022.
- [Press et al., 2023] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, dan Mike Lewis. Mengukur dan mempersempit kesenjangan komposisi dalam model bahasa. Dalam Temuan Asosiasi Linguistik Komputasional: EMNLP 2023, halaman 5687–5711, 2023.
- [Pryzant dkk., 2023] Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, dan Michael Zeng. Pengoptimalan prompt otomatis dengan "gradient descent" dan pencarian berkas. Dalam Konferensi 2023 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, 2023.
- [Qiu dkk., 2020] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, dan Xuanjing Huang. Model pra-latihan untuk pemrosesan bahasa alami: Sebuah survei. Sains China Technological Sciences, 63(10):1872–1897, 2020.
- [Radford dkk., 2018] Alec Radford, Karthik Narasimhan, Tim Salimans, dan Ilya Sutskever. Meningkatkan pemahaman bahasa melalui pelatihan awal generatif. OpenAI, 2018.
- [Radford dkk., 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, dan Ilya Sutskever. Model bahasa adalah pembelajar multitugas tanpa pengawasan. Blog OpenAI, 1(8), 2019.
- [Radford dkk., 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, dan Ilya Sutskever. Mempelajari model visual yang dapat ditransfer dari supervisi bahasa alami. Dalam konferensi internasional tentang pembelajaran mesin, halaman 8748–8763. PMLR, 2021.
- [Rae dkk., 2019] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, dan Timothy P Lillicrap. Transformator kompresif untuk pemodelan urutan jarak jauh. Dalam Konferensi Internasional tentang Representasi Pembelajaran, 2019.
- [Rafailov dkk., 2024] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, dan Chelsea Finn. Optimasi preferensi langsung: Model bahasa Anda secara diam-diam merupakan model penghargaan. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 36, 2024.
- [Raffel dkk., 2020] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, dan Peter J. Liu. Menjelajahi batas-batas pembelajaran transfer dengan transformator teks-ke-teks terpadu. Jurnal Penelitian Pembelajaran Mesin, 21(140):1–67, 2020.
- [Ramachandran dkk., 2017] Prajit Ramachandran, Barret Zoph, dan Quoc V Le. Pencarian fungsi aktivasi. arXiv preprint arXiv:1710.05941, 2017.
- [Rolnick dkk., 2019] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, dan Gregory Wayne. Pengalaman pemutaran ulang untuk pembelajaran berkelanjutan. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 32, 2019.
- [Rosenfeld dkk., 2020] Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, dan Nir Shavit. Prediksi konstruktif kesalahan generalisasi lintas skala. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2020.
- [Ruan dkk., 2024] Junhao Ruan, Long Meng, Weiqiao Shan, Tong Xiao, dan Jingbo Zhu. Sebuah survei llm survei. <https://github.com/NiuTrans/ABigSurveyOfLLMs>, Tahun 2024.
- [Rubin dkk., 2022] Ohad Rubin, Jonathan Herzig, dan Jonathan Berant. Belajar mengambil perintah untuk pembelajaran dalam konteks. Dalam Prosiding Konferensi 2022 North American Chapter of the Association for Computational Linguistics: Human Language Technologies, halaman 2655–2671, 2022.
- [Russell, 2019] Stuart Russell. Kecocokan Manusia: Kecerdasan Buatan dan Masalah Kontrol. Viking, 2019.
- [Sanh dkk., 2020] Victor Sanh, Thomas Wolf, dan Alexander Rush. Pemangkasan gerakan: Kelangkaan adaptif melalui penyempurnaan. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 33:20378–20389, 2020.

- [Sanh dkk., 2022] Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Bider-man, Leo Gao, Thomas Wolf, dan Alexander M Rush. Pelatihan yang didorong oleh multitask memungkinkan generalisasi tugas zero-shot. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2022.
- [Schick dkk., 2023] Timo Schick, Jane A. Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izac-ard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, dan Sebastian Riedel. PEER: Model bahasa kolaboratif. Dalam Prosiding Konferensi Internasional Kesebelas tentang Representasi Pembelajaran, 2023.
- [Schick dkk., 2024] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, dan Thomas Scialom. Toolformer: Model bahasa dapat belajar sendiri untuk menggunakan alat. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 36, 2024.
- [Schmidhuber, 2015] Jürgen Schmidhuber. Pembelajaran mendalam dalam jaringan saraf: Tinjauan umum. Jaringan saraf, 61:85–117, 2015.
- [Schulman et al., 2015] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, dan Pieter Abbeel. Optimalisasi kebijakan wilayah kepercayaan. Dalam Prosiding Konferensi Internasional ke-32 tentang Pembelajaran Mesin-Volume 37, halaman 1889–1897, 2015.
- [Schulman dkk., 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, dan Oleg Klimov. Algoritma optimasi kebijakan proksimal. arXiv pracetak arXiv:1707.06347, 2017.
- [Sennrich dkk., 2016] Rico Sennrich, Barry Haddow, dan Alexandra Birch. Meningkatkan model terjemahan mesin neural dengan data monolingual. Dalam Prosiding Pertemuan Tahunan ke-54 Asosiasi Linguistik Komputasional (Volume 1: Makalah Panjang), halaman 86–96, 2016.
- [Seo et al., 2017] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, dan Hannaneh Hajishirzi. Aliran perhatian dua arah untuk pemahaman mesin. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2017.
- [Shannon, 1951] Claude E Shannon. Prediksi dan entropi bahasa Inggris tercetak. Sistem Bell teknis jurnal, 30(1):50–64, 1951.
- [Shaw dkk., 2018] Peter Shaw, Jakob Uszkoreit, dan Ashish Vaswani. Perhatian diri dengan representasi posisi relatif. Dalam Prosiding Konferensi 2018 North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Makalah Pendek), halaman 464–468, 2018.
- [Shazeer, 2019] Noam Shazeer. Dekode transformator cepat: Anda hanya perlu satu kepala tulis. arXiv preprint arXiv:1911.02150, 2019.
- [Shazeer, 2020] Noam Shazeer. Varian Glu meningkatkan transformator. arXiv preprint arXiv:2002.05202, 2020.
- [Shen dkk., 2020] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, dan Kurt Keutzer. Q-bert: Kuantisasi presisi sangat rendah berbasis Hessian dari bert. Dalam Prosiding Konferensi AAAI tentang Kecerdasan Buatan, volume 34, halaman 8815–8821, 2020.
- [Shoeybi dkk., 2019] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, dan Bryan Catanzaro. Megatron-Im: Melatih model bahasa parameter multi-miliar menggunakan paralelisme model. arXiv pracetak arXiv:1909.08053, 2019.
- [Skalse dkk., 2022] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, dan David Krueger. Mendefinisikan dan mengkarakterisasi permainan berhadiah. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 35:9460–9471, 2022.
- [Snell et al., 2022] Charlie Snell, Dan Klein, dan Ruiqi Zhong. Belajar dengan menyaring konteks. arXiv

pracetak arXiv:2209.15189, 2022.

[Socher dkk., 2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, dan Christopher Potts. Model mendalam rekursif untuk komposisionalitas semantik pada kumpulan pohon sentimen. Dalam Prosiding konferensi 2013 tentang metode empiris dalam pemrosesan bahasa alami, halaman 1631–1642, 2013.

[Song dkk., 2019] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, dan Tie-Yan Liu. Mass: Pra-pelatihan urutan ke urutan yang disamarkan untuk pembuatan bahasa. Dalam Konferensi Internasional tentang Pembelajaran Mesin, halaman 5926–5936. PMLR, 2019.

[Stiennon dkk., 2020] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, dan Paul F Christiano. Belajar meringkas dengan umpan balik manusia. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 33:3008–3021, 2020.

[Su et al., 2024] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, dan Yunfeng Liu. Ro-former: Transformator yang ditingkatkan dengan penyematkan posisi putar. Komputasi saraf, 568:127063, 2024.

[Sun et al., 2020] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, dan Denny Zhou. Mobilebert: perangkat lunak kompak yang tidak bergantung pada tugas untuk perangkat dengan sumber daya terbatas. Dalam Prosiding Pertemuan Tahunan ke-58 Asosiasi Linguistik Komputasional, halaman 2158–2170, 2020.

[Sutskever et al., 2014] Ilya Sutskever, Oriol Vinyals, dan Quoc V Le. Pembelajaran urutan ke urutan dengan jaringan saraf. Kemajuan dalam sistem pemrosesan informasi saraf, 27, 2014.

[Sutton dan Barto, 2018] Richard S. Sutton dan Andrew G. Barto. Pembelajaran Penguatan: Pengantar Jurnal Manajemen dan Kewirausahaan (edisi ke-2). MIT Press, 2018.

[Szepesvári, 2010] Csaba Szepesvári. Algoritma untuk pembelajaran penguatan. Kuliah Sintesis pada Arti-Kecerdasan Buatan dan Pembelajaran Mesin, 4(1):1–103, 2010.

[Talmor dan Berant, 2018] Alon Talmor dan Jonathan Berant. Web sebagai basis pengetahuan untuk menjawab pertanyaan kompleks. arXiv pracetak arXiv:1803.06643, 2018.

[Taori dkk., 2023] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, dan Tatsunori B. Hashimoto. Stanford alpaca: Model llama yang mengikuti instruksi. https://github.com/tatsu-lab/stanford_alpaca, Tahun 2023.

[Tay et al., 2020] Yi Tay, Mostafa Dehghani, Dara Bahri, dan Donald Metzler. Transformator yang efisien: Sebuah studi survei. CoRR, abs/2009.06732, 2020.

[Team dkk., 2024] Tim Gemma, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, dkk. Gemma 2: Meningkatkan model bahasa terbuka dalam ukuran praktis. Pracetak arXiv arXiv:2408.00118, 2024.

[Teknium, 2023] Teknium. Openhermes 2.5: Kumpulan data sintetis terbuka untuk asisten llm generalis, 2023. URL <https://huggingface.co/datasets/teknium/OpenHermes-2.5>.

[Touvron dkk., 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Ro-driguez, Armand Joulin, Edouard Grave, dan Guillaume Lample. Llama: Model bahasa dasar yang terbuka dan efisien. arXiv pracetak arXiv:2302.13971, 2023a.

[Touvron dkk., 2023] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernan-des, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Median Khabsa, Is-abel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schel-ten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor,

Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, dan Thomas Scialom. Llama 2: Landasan terbuka dan model obrolan yang disempurnakan. arXiv pracetak arXiv:2307.09288, 2023b.

[Uesato dkk., 2022] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, dan Irina Higgins. Memecahkan soal cerita matematika dengan umpan balik berbasis proses dan hasil. arXiv preprint arXiv:2211.14275, 2022.

[Vaswani dkk., 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, dan Illia Polosukhin. Hanya perhatian yang Anda butuhkan. Dalam Prosiding Kemajuan Sistem Pengolahan Informasi Neural, volume 30, 2017.

[Von Oswald dkk., 2023] Johannes Von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, dan Max Vladymyrov. Transformer belajar dalam konteks melalui penurunan gradien. Dalam Prosiding Konferensi Internasional tentang Pembelajaran Mesin, halaman 35151–35174. PMLR, 2023.

[Wang dkk., 2024] Chenglong Wang, Hang Zhou, Yimin Hu, Yifu Huo, Bei Li, Tongran Liu, Tong Xiao, dan Jingbo Zhu. Esrl: Pembelajaran penguatan berbasis pengambilan sampel yang efisien untuk pembuatan urutan. Dalam Prosiding Konferensi AAAI tentang Kecerdasan Buatan, halaman 19107–19115, 2024.

[Wang dkk., 2023] Liyuan Wang, Xingxing Zhang, Hang Su, dan Jun Zhu. Survei komprehensif tentang pembelajaran berkelanjutan: Teori, metode, dan aplikasi. arXiv preprint arXiv:2302.00487, 2023a.

[Wang dkk., 2019] Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, dan Lidia S Chao. Mempelajari model transformator mendalam untuk penerjemahan mesin. Dalam Prosiding Pertemuan Tahunan ke-57 Asosiasi Linguistik Komputasional, halaman 1810–1822, 2019.

[Wang dkk., 2022] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, dan Denny Zhou. Ensemble yang ditambah dengan alasan dalam model bahasa. Pracetak arXiv arXiv:2207.00747, 2022a.

[Wang dkk., 2023] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, dan Denny Zhou. Konsistensi diri meningkatkan penalaran rantai pemikiran dalam model bahasa. Dalam Prosiding Konferensi Internasional Kesebelas tentang Representasi Pembelajaran, 2023b.

[Wang dkk., 2022] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Es-haan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mi-rali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, dan Xudong Shen. Super-naturalinstructions: Generalisasi melalui instruksi deklaratif pada 1600+ tugas nlp. Dalam Prosiding Konferensi 2022 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 5085–5109, 2022b.

[Wang dkk., 2023] Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khy-athi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, dan Hannaneh Hajishirzi. Seberapa jauh unta dapat terbang? Menjelajahi status penyetelan instruksi pada sumber daya terbuka. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 36:74764–74786, 2023c.

[Wang dkk., 2023] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, dan Hannaneh Hajishirzi. Belajar mandiri: Menyelaraskan model bahasa dengan instruksi yang dibuat sendiri. Dalam Prosiding Pertemuan Tahunan ke-61 Asosiasi Linguistik Komputasional (Volume 1: Makalah Panjang), halaman 13484–13508, 2023d.

[Wang dkk., 2023] Zhenyi Wang, Enneng Yang, Li Shen, dan Heng Huang. Sebuah survei komprehensif tentang lupa dalam pembelajaran mendalam di luar pembelajaran berkelanjutan. arXiv pracetak arXiv:2307.09218, 2023e.

- [Warstadt dkk., 2019] Alex Warstadt, Amanpreet Singh, dan Samuel R Bowman. Penilaian penerimaan jaringan saraf. *Transaksi Asosiasi Linguistik Komputasional*, 7:625–641, 2019.
- [Wei et al., 2022] Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, dan Quoc V Le. Model bahasa yang disempurnakan adalah pembelajar tanpa peluang. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2022a.
- [Wei et al., 2022] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, dan William Fedus. Kemampuan yang muncul dari model bahasa besar. arXiv pracetak arXiv:2206.07682, 2022b.
- [Wei dkk., 2022] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, dan Denny Zhou. Dorongan berantai pemikiran memunculkan penalaran dalam model bahasa yang besar. *Kemajuan dalam Sistem Pemrosesan Informasi Neural*, 35:24824–24837, 2022c.
- [Welleck dkk., 2023] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, dan Yejin Choi. Menghasilkan urutan dengan belajar mengoreksi diri sendiri. Dalam Prosiding Konferensi Internasional Kesebelas tentang Representasi Pembelajaran, 2023.
- [Weng, 2021] Lilian Weng. Bagaimana cara melatih model yang sangat besar pada banyak GPU? lilianweng.github.io, Sep 2021. URL <https://lilianweng.github.io/posts/2021-09-25-train-large/>.
- [Wiener, 1960] Norbert Wiener. Beberapa konsekuensi moral dan teknis dari otomatisasi: Saat mesin belajar, mereka mungkin mengembangkan strategi yang tak terduga dengan kecepatan yang membingungkan programmer. *Sains*, 131(3410): 1355–1358, 1960.
- [Williams dkk., 2018] Adina Williams, Nikita Nangia, dan Samuel Bowman. Korpus tantangan dengan cakupan luas untuk pemahaman kalimat melalui inferensi. Dalam Prosiding Konferensi 2018 North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Makalah Panjang), halaman 1112–1122, 2018.
- [Williams, 1992] Ronald J Williams. Algoritma mengikuti gradien statistik sederhana untuk koneksiisme pembelajaran penguatan. *Pembelajaran mesin*, 8:229–256, 1992.
- [Wingate dkk., 2022] David Wingate, Mohammad Shoeybi, dan Taylor Sorensen. Kompresi cepat dan pengkondisian kontrastif untuk pengendalian dan pengurangan toksitas dalam model bahasa. Dalam Temuan Asosiasi Linguistik Komputasional: EMNLP 2022, halaman 5621–5634, 2022.
- [Wu et al., 2024] Wilson Wu, John X Morris, dan Lionel Levine. Apakah model bahasa merencanakan token masa depan? arXiv preprint arXiv:2404.00859, 2024.
- [Wu dkk., 2021] Yuhuai Wu, Markus Norman Rabe, DeLesley Hutchins, dan Christian Szegedy. Transformator hafalan. Dalam Prosiding Konferensi Internasional tentang Pembelajaran Representasi, 2021.
- [Wu et al., 2023] Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, dan Hannaneh Hajishirzi. Umpan balik manusia yang terperinci memberikan hasil yang lebih baik untuk pelatihan model bahasa. Dalam Konferensi ke-37 tentang Sistem Pemrosesan Informasi Neural, 2023.
- [Xia et al., 2024] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, dan Danqi Chen. Kurang: Memilih data yang berpengaruh untuk penyetelan instruksi yang ditargetkan. Pracetak arXiv arXiv:2402.04333, 2024.
- [Xiao dkk., 2024] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, dan Mike Lewis. Model bahasa streaming yang efisien dengan attention sink. Dalam Prosiding Konferensi Internasional Kedua Belas tentang Representasi Pembelajaran, 2024.
- [Xiao dan Zhu, 2023] Tong Xiao dan Jingbo Zhu. Pengantar transformator: perspektif nlp. arXiv pracetak arXiv:2311.17633, 2023.
- [Xiao dkk., 2013] Tong Xiao, Jingbo Zhu, dan Tongran Liu. Mengantongi dan meningkatkan mesin statistik sistem penerjemahan. *Kecerdasan Buatan*, 195:496–527, 2013.

- [Xiao dkk., 2019] Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, dan Tongran Liu. Berbagi bobot perhatian untuk transformator cepat. Dalam Prosiding Konferensi Gabungan Internasional Ke-28 tentang Kecerdasan Buatan (IJCAI-19), halaman 5292–5298, 2019.
- [Xie dkk., 2022] Sang Michael Xie, Aditi Raghunathan, Percy Liang, dan Tengyu Ma. Penjelasan tentang pembelajaran dalam konteks sebagai inferensi Bayesian implisit. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2022.
- [Xin dkk., 2020] Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, dan Jimmy Lin. Deebert: Keluar lebih awal secara dinamis untuk mempercepat inferensi bert. Dalam Prosiding Pertemuan Tahunan ke-58 Asosiasi Linguistik Komputasional, halaman 2246–2251, 2020.
- [Xu dkk., 2024] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, dan Daxin Jiang. WizardLM: Memberdayakan model bahasa besar yang telah dilatih sebelumnya untuk mengikuti instruksi yang kompleks. Dalam Konferensi Internasional Keduabelas tentang Representasi Pembelajaran, 2024.
- [Yang dkk., 2024] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, dkk. Qwen2. 5 laporan teknis. arXiv pracetak arXiv:2412.15115, 2024.
- [Yang et al., 2019] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, dan Quoc V Le. Xlnet: Pra-pelatihan autoregresif umum untuk pemahaman bahasa. Kemajuan dalam sistem pemrosesan informasi saraf, 32, 2019.
- [Yao dkk., 2024] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, dan Karthik Narasimhan. Pohon pemikiran: Pemecahan masalah yang disengaja dengan model bahasa yang besar. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 36, 2024.
- [Yarowsky, 1995] David Yarowsky. Disambiguasi makna kata tanpa pengawasan yang menyaingi metode pengawasan. Dalam Prosiding pertemuan tahunan ke-33 asosiasi linguistik komputasional, halaman 189–196, 1995.
- [Yu dkk., 2023] Zihan Yu, Liang He, Zhen Wu, Xinyu Dai, dan Jiajun Chen. Menuju rantai-of-strategi pemicu pemikiran: Sebuah survei. arXiv preprint arXiv:2310.04959, 2023.
- [Zaheer dkk., 2020] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, C. Alberti, S. Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, L. Yang, dan A. Ahmed. Big bird: Transformer untuk urutan yang lebih panjang. Kemajuan dalam sistem pemrosesan informasi saraf, 33:17283–17297, 2020.
- [Zellers dkk., 2018] Rowan Zellers, Yonatan Bisk, Roy Schwartz, dan Yejin Choi. Swag: Kumpulan data adversarial berskala besar untuk inferensi akal sehat yang berlandaskan. Dalam Prosiding Konferensi 2018 tentang Metode Empiris dalam Pemrosesan Bahasa Alami, halaman 93–104, 2018.
- [Zhang dan Sennrich, 2019] Biao Zhang dan Rico Sennrich. Normalisasi lapisan akar kuadrat rata-rata. inovasi dalam Sistem Pemrosesan Informasi Neural, 32, 2019.
- [Zhang dkk., 2023] Zhuosheng Zhang, Yao Yao, Aston Zhang, Xiangru Tang, Xinbei Ma, Zhiwei He, Yiming Wang, Mark Gerstein, Rui Wang, Gongshen Liu, dan Hai Zhao. Menyalakan kecerdasan bahasa: Panduan penumpang dari penalaran rantai pemikiran hingga agen bahasa. arXiv pracetak arXiv:2311.11797, 2023a.
- [Zhang dkk., 2023] Zhuosheng Zhang, Aston Zhang, Mu Li, dan Alex Smola. Rangkaian pemikiran otomatis yang mendorong dalam model bahasa besar. Dalam Konferensi Internasional Kesebelas tentang Pembelajaran Representasi, 2023b.
- [Zhao dkk., 2024] Hao Zhao, Maksym Andriushchenko, Francesco Croce, dan Nicolas Flammarion. Panjang lebih cocok untuk penyelarasian: Dasar yang sederhana tetapi sulit dikalahkan untuk penyempurnaan instruksi. Pracetak arXiv arXiv:2402.04833, 2024.
- [Zhao dkk., 2023] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Z. Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jianyun Nie, dan Ji rong Wen.

Survei model bahasa besar. arXiv pracetak arXiv:2303.18223, 2023.

[Zhou dkk., 2023] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivas Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, dan Omer Levy. Lima: Lebih sedikit lebih baik untuk penyelarasan. arXiv pracetak arXiv:2305.11206, 2023a.

[Zhou dkk., 2023] Denny Zhou, Nathanael Schärlí, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, dan Ed H. Chi. Dorongan paling sedikit hingga paling banyak memungkinkan penalaran kompleks dalam model bahasa yang besar. Dalam Prosiding Konferensi Internasional Kesebelas tentang Representasi Pembelajaran, 2023b.

[Zhou dkk., 2020] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, dan Furu Wei. Bert kehilangan kesabaran: Inferensi cepat dan kuat dengan keluar lebih awal. Kemajuan dalam Sistem Pemrosesan Informasi Neural, 33:18330–18341, 2020.

[Zhou dkk., 2023] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, dan Jimmy Ba. Model bahasa besar adalah insinyur perintah tingkat manusia. Dalam Konferensi Internasional Kesebelas tentang Representasi Pembelajaran, 2023c.

[Zoph dan Le, 2016] Barret Zoph dan Quoc Le. Pencarian arsitektur neural dengan pembelajaran penguatan. Dalam Prosiding Konferensi Internasional tentang Representasi Pembelajaran, 2016.

[Zoph dkk., 2020] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, dan Quoc Le. Memikirkan kembali pra-pelatihan dan pelatihan mandiri. Kemajuan dalam sistem pemrosesan informasi saraf, 33:3833–3845, 2020.

Indeks

- k-NN, 74 k-
- NN LM, 76
- pemodelan bahasa k-NN, 76 k-
- tetangga terdekat, 74
- A2C, 175
- fungsi nilai tindakan, 171
- keuntungan, 175
- aktor-kritikus keuntungan, 175
- Agen, 47
- ALiBi, 85
- penyelarasan,
- 46 perhatian dengan bias linier, 85
- pembelajaran mesin otomatis, 137
- desain prompt otomatis, 137
- AutoML, 137
- agen otonom, 134
- BART, 19
- BERT, 1
- Pengambilan sampel terbaik dari N , 197
- Pengambilan sampel BoN, 197
- Model Bradley Terry, 178
- anotasi kalkulasi, 113 lupa yang fatal, 34 pemodelan bahasa kausal, 9 rangkaian pemikiran, 113 dorongan rangkaian pemikiran, 53 penyelesaian, 6 generalisasi komposisi, 122 CoT, 113 Dorongan COT, 53 model bahasa lintas bahasa, 28 hadiah kumulatif, 172
- sengaja-lalu-hasilkan, 126 demonstrasi, 6 optimasi preferensi langsung, 190 Rotasi Dokumen, 20 DPO, 190 DTG, 126
- kemampuan yang muncul, 63 memori eksternal, 74
- Ekstrapolasi, 81
- beberapa bidikan COT yang mendorong, 54
- unit linier berpagar, unit linier gaussian error 58 , 58
- GeLU, 58
- GLU, 58
- GPT, 1
- GQA, 80
- Perhatian kueri yang dikelompokkan, 80
- prompt keras, 140
- penyelarasan preferensi manusia, 152
- Liga Inggris, 53
- ICT, 6
- pengambilan sampel penting, 180 pembelajaran dalam konteks, 6, 53, 95 inversi masukan, 163 penyelarasan instruksi, 152 penyempurnaan instruksi, 43, 154
- interferensi, 30 memori internal, 74
- Interpolasi, 82
- kesalahan tak tereduksi, 63
- cache nilai kunci, 68
- Kapasitas KV, 68
- pemetaan label, 105
- Belajar dari Umpam Balik Manusia, 47
- dorongan paling sedikit hingga paling banyak, 119 LLM konteks panjang, 66
- pemodelan bahasa bertopeng, 1, 9
- mBERT, 28
- metode berbasis memori, 74
- MQA, 79
- BERT multibahasa, 28
- perhatian multi-query, 79
- NAS, 137
- pencarian arsitektur saraf, 137
- prediksi kalimat berikutnya, 13
- NSP, 13
- pembelajaran penguatan offline, 193
- prompt COT satu kali, 54
- Pendekatan Berbasis Hasil, 195

- masalah optimasi berlebihan, 189
- Estimasi Kinerja, 137 fungsi kinerja, 172 kesenjangan kinerja dipulihkan, 167 pemodelan bahasa yang diubah, 11 PGR, 167 Model Plackett-Luce, 184 PPO, 50, 181 penyempurnaan awalan, 144 pemodelan bahasa awalan, 16 dekomposisi masalah, 116 Pendekatan Berbasis Proses, 195 penyematan prompt, 148 rekayasa prompt, 95 pengoptimalan prompt, 137 Prompt Search Space, 137 rekayasa prompt, 51 optimasi kebijakan proksimal, 50, 181
- Fungsi nilai Q, 171
- RAG, 76 fungsi rasio, 180 unit linier yang diperbaiki, 58 pembelajaran penguatan dari umpan balik manusia, 47, 153 pengambilan sampel penolakan, 198 ekstraksi relasi, 108 ReLU, 58 pembangkitan yang ditingkatkan pengambilannya, 76 pengembalian, 172 permainan hadiah, 189 peretasan hadiah, 189 Model Hadiah, 47 RLHF, 47, 153 Roberta, 26 tahun
- contoh efisien, 164 hukum penskalaan, 63 konsistensi diri, 129 instruksi mandiri, 160 pembelajaran mandiri, 3 pelatihan mandiri, 3 Penataan Ulang Kalimat, 19 Model Pengkodean Urutan, 3 Model Pembuatan Urutan, 3 SFT, 47, 152
- prediksi satu putaran, 155 prompt lunak, 140 Span Masking, fungsi nilai status 19, 171 Kinerja Plafon Kuat, 167 Pembuatan Sub-Masalah, 118 Pemecahan Sub-Masalah, 118 Hipotesis Keselarasan Superfisial, 164 Penyetelan Halus yang Diawasi, 47 penyetelan halus yang diawasi, 152 pembelajaran yang diawasi, 2 tujuan pengganti, 180
- T5, 15 TD, 176 perbedaan temporal, 176 pelengkapan teks, 109 transformasi teks, 109 Penghapusan Token, 19 Penyamaran Token, 19 Transformer, 1 pemodelan bahasa terjemahan, 29 wilayah kepercayaan, 181
- pembelajaran tanpa pengawasan, 2 Kinerja Lemah, 167 generalisasi lemah ke kuat, 166 Kinerja Lemah-ke-Kuat, 167
- XLM, 28
- COT tanpa tembakan, 54 pembelajaran tanpa tembakan, 45