Perkembangan Teknik Pembelajaran *Artificial Neural Network* hingga *Transformer* dalam Representasi Kata Lintas Bahasa

Ilham Firdausi Putra
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung, Indonesia
ilhamfputra31@gmail.com

***Ringkasan*—Representasi kata adalah hal yang sangat penting dalam bidang pemrosesan bahasa. Dalam tugas ini, dieksplorasi perkembangan teknik pembelajaran *Artificial Neural Network* hingga *Transformer* dalam representasi teks lintas bahasa. Tidak hanya itu, juga akan dilihat penggunaanya pada perkembangan representasi teks lintas bahasa dari *monolingual mapping* hingga *joint optimization*.**

***Index Terms*—artificial neural network, transformer, representasi teks lintas bahasa**

I. Pendahuluan

Sebagian besar algoritma pembelajaran mesin memerlukan representasi teks dalam bentuk numerik. Untuk hal itu, berkembanglah berbagai teknik untuk merepresentasikan teks sebaik mungkin. Bentuk paling sederhananya adalah pendekatan *one-hot-vector* yang merepresentasikan teks berdasarkan ada atau tidaknya saja. Representasi seperti ini memiliki kekurangan seperti tidak diperhatikannya letak kata dan membesarnya representasi kata seiring membesarnya kosa kata.

Kekurangan dari representasi *one-hot-ector* dapat diselesaikan dengan *word embedding* seperti Word2vec [1] yang mempelajari representasi kata sebagai vektor bernilai riil. Pembangunan *word embedding* seperti Word2vec memanfaatkan *Artificial Neural Network* dengan teknik seperti *Continuous Bag of Words* (CBOW) dan Skip-gram.

Representasi teks dengan *word embedding* masih memiliki kelemahan berupa masih dangkalnya representasi. Representasi *word embedding* tidak dapat menangkap interaksi antar kata di kalimat yang kompleks. Oleh karena itu, berkembanglah *language model* seperti XLM [2] yang tidak hanya belajar di level kata, tetapi sampai dapat memperhatikan konteks di mana kata tersebut berada. Pembangunan *language model* seperti XLM memanfaatkan arsitektur *transformer* yang lebih anyar.

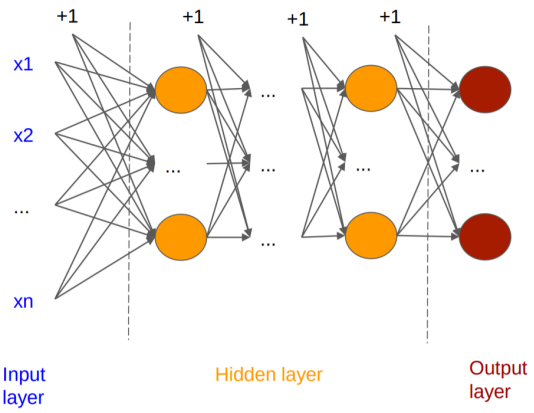
Dalam tugas ini akan kita eksplorasi perkembangan penggunaan *Artificial Neural Network* dalam pembangunan *word embedding* lintas bahasa pada Word2vec. Setelah itu kita lihat juga hubungannya ke perkembangan arsitektur *transformer* dalam pembangunan *language model* lintas bahasa seperti XLM. Kita juga akan lihat bagaimana perkembangan representasi lintas bahasa berkembang dari *mapping* secara linear di Word2vec, hingga penggunaan *sub-word* di XLM.

II. Artificial Neural Network

Teknik pembelajaran *Artificial Neural Network* (ANN), yang termasuk jenis ***discriminative***, pertama kali dimodelkan pada tahun 1943 oleh Warren McCulloch dan Walter Pits [3]. Pemodelan ini terinspirasi dari cara neuron bekerja di otak. Semenjak itu, berkembang bebagai kemajuan dalam metode dan teknik melakukan pembelajaran dengan ANN. Keefektifan ANN dibuktikan dengan performanya yang berhasil mengalahkan teknik non-ANN di berbagai bidang [4] [5].

A. Teknik Pembelajaran

Sebuah ANN terdiri dari lapisan masukkan (*input*), lapisan tersembunyi (*hidden*), dan lapisan keluaran (*output*) seperti dapat dilihat pada Gambar 1. Tiap garis yang menghubungkan neuron memiliki bobot yang dapat diperbaharui. Dalam pembelajaran, ANN akan melewati fase yang disebut dengan *forward propagation*, *backpropagation*, dan pembaharuan bobot.

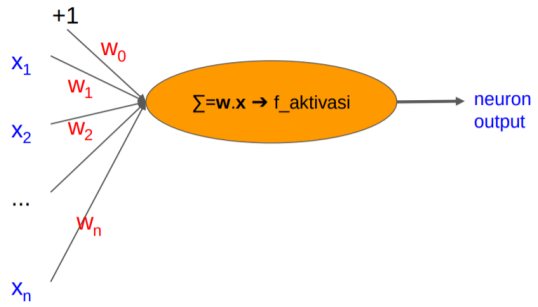


Gambar 1. Ilustrasi ANN [6].

Pada fase *forward propagation*, tiap input numerik diproses sesuai dengan bobot dari tiap neuronnya. Output suatu neuron akan menjadi input dari neuron pada lapisan setelahnya. Pada Gambar 2, diberikan input numerik riil berupa $X_1, X_2, \ldots, X_n$ ($X_0 = 1$ sebagai bias), bobot

$W_0, W_1, W_2, \ldots, W_n$, dan sebuah fungsi aktifasi f. Maka output dikomputasi dengan hitungan pada 1.

$$output = f(\sum_{i=0}^{n} x_i w_i) \quad (1)$$

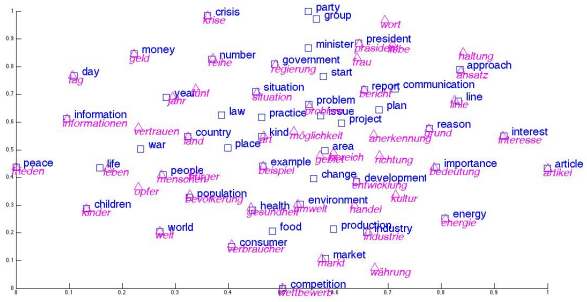


Gambar 2. Ilustrasi suatu neuron [6].

Komputasi output dilakukan hingga lapisan terakhir. Setelah output didapatkan, maka pembelajaran masuk fase *backpropagation*. Pada fase ini kesalahan antara output prediksi dihitung dengan output seharusnya dan diteruskan kebelakang. Melalui error yang sudah dikomputasi ini kemudia bobot diperbaharui pada fase terakhir. Dengan melakukan proses ini terus menerus hingga diperoleh *error* yang minimum, dapat diperoleh algoritme yang mangkus mengaproksimasi fungsi aslinya.

B. Penggunaan ANN di Word2Vec dan Representasi Teks Lintas Bahasa

Salah satu bentuk dari Transfer Learning adalah pembelajaran lintas bahasa [7]. Pada pembelajaran lintas bahasa, pembelajaran pertama-tama dilakukan di bahasa lain yang lebih populer dan kemudian digunakan pada bahasa lain yang lebih tidak populer. Untuk dapat melakukan hal ini, diperlukan representasi bahasa pada ruang yang sama seperti dapat dilihat pada ilustrasi di gambar 3.



Gambar 3. Ilustrasi ruang *embedding* antara dua bahasa [8].

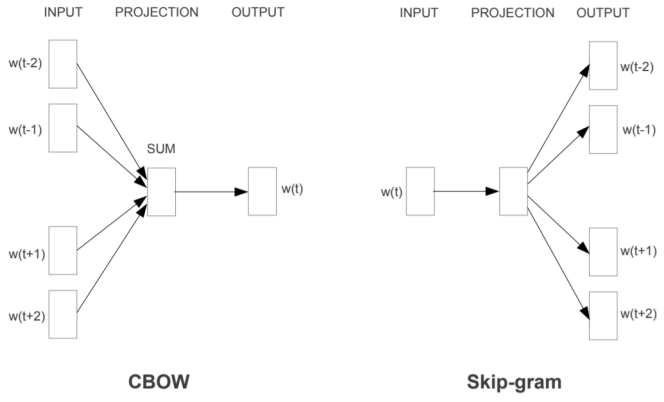
Sebelumnya, diperlukan ahli untuk membangun kamus kata-kata dan kalimat secara manual untuk dapat membandingkan kata dari dua bahsa. Seiring dengan perkembangan teknologi, berbagai teknik berkembang untuk dapat melakukan hal ini secara otomatis. Berdasarkan [9], secara garis besar terdapat 2 pendekatan berbeda dalam membangun ruang *embedding* antar bahasa:

1) *Monolingual mapping*: Pada teknik ini, model dilatih secara independen pada bahasa masing-masing. Kemudian mapping antara bahasa dipelajari untuk mendapatkan representasi antar bahasa.
2) *Joint optimization*: Pada teknik ini, model dilatih pada korpus antar bahasa. Model kemudian mengoptimisasi kombinasi dari monolingual dan cross-lingual loss.

Pada *monolingual mapping*, salah satu teknik adalah dengan pertama-tama mempelajari representasi bahasa menggunakan model Skip-gram atau Continuous Bag-of-Words (CBOW) yang didistribusikan diusulkan oleh [10]. Model-model ini mempelajari representasi kata menggunakan arsitektur *neural network* sederhana yang bertujuan untuk memprediksi tetangga kata. Karena kesederhanaannya, model Skip-gram dan CBOW dapat dilatih pada sejumlah besar data teks. Pada implementasi paralelnya model ini dapat belajar dari miliaran kata dalam hitungan jam.

Baru-baru ini ditunjukkan bahwa representasi kata yang didistribusikan secara mengejutkan menangkap banyak keteraturan linguistik, dan ada banyak jenis kesamaan di antara kata-kata yang dapat dinyatakan sebagai terjemahan linear [11]. Misalnya, operasi vektor ”raja” - ”pria” + ”wanita” menghasilkan vektor yang dekat dengan ”ratu”.

Dua model khusus untuk mempelajari representasi kata yang dapat dilatih secara efisien pada banyak data teks adalah model Skip-gram dan CBOW yang diperkenalkan di [10]. Dalam model CBOW, tujuan pelatihannya adalah untuk menggabungkan representasi kata di sekitarnya untuk memprediksi kata di tengah. Sedangkan dalam model Skip-gram, tujuan pelatihan adalah untuk mempelajari representasi kata vektor yang pandai memprediksi konteksnya dalam kalimat yang sama [10]. Model arsitektur dari dua metode ini ditunjukkan pada Gambar 4.



Gambar 4. Dalam model CBOW, representasi konteks terdistribusi (atau kata-kata sekitarnya) digabungkan untuk memprediksi kata di tengah. Dalam model Skip-gram, representasi terdistribusi dari kata input digunakan untuk memprediksi kata-kata sekitarnya [12].

Lebih formal, diberi urutan kata pelatihan

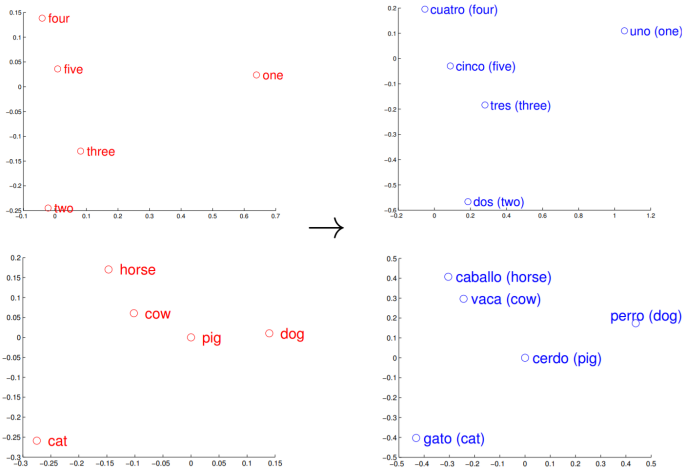
$w1, w2, w3, ..., wT$, tujuan dari model Skip-gram adalah untuk memaksimalkan probabilitas log rata-rata

$$\frac{1}{T}\sum_{t=1}^{T}\left[\sum_{j=-k}^{K}\log p(w_{t+j}|w_t)\right] \quad (2)$$

di mana k adalah ukuran jendela pelatihan. Iterasi penjumlahan di bagian dalam berjalan dari $-k$ ke k untuk menghitung probabilitas log benarnya memprediksi kata w_{t+j} jika kata di tengah w_t. Iterasi penjumlahan di luar mencakup semua kata dalam korpus pelatihan.

Ketika dilatih tentang dataset besar, model Skip-gram atau CBOW ini menangkap banyak informasi semantik. Seperti yang disebutkan sebelumnya, kata-kata yang berkaitan erat memiliki representasi vektor yang serupa, misalnya, sekolah dan universitas, danau, dan sungai. Ini karena sekolah dan universitas muncul dalam konteks yang sama, sehingga selama pelatihan representasi vektor dari kata-kata ini didorong untuk menjadi dekat satu sama lain. Lebih menarik lagi, vektor menangkap hubungan antara konsep melalui operasi linier. Misalnya, $vektor(Prancis) - vektor(Paris)$ mirip dengan $vektor(Italia) - vektor(Roma)$.

Pada gambar 5, dapat dilihat visualisasi hasil pembelajaran Skip-gram atau CBOW. Gambar 5 memvisualisasikan vektor untuk angka dan hewan dalam bahasa Inggris dan Spanyol, dan dapat dengan mudah dilihat bahwa konsep-konsep ini memiliki susunan geometris yang serupa. Hal ini dikarenakan semua bahasa umum memiliki konsep yang didasarkan pada dunia nyata (seperti kucing adalah binatang yang lebih kecil dari seekor anjing), sering kali ada kesamaan kuat antara ruang vektor. Kesamaan pengaturan geometris dalam ruang vektor adalah alasan utama mengapa metode ini dapat bekerja dengan baik.



Gambar 5. Vektor yang sudah dipelajari diproyeksikan ke ruang dua dimensi menggunakan PCA dan dirotasi.

Dikarenakan miripnya representasi bahasa, jika kita mengetahui translasi beberapa objek, contoh angka satu sampai lima, kita dapat menyesuaikan representasi antar bahasa dengan rotasi, *scaling*, dan translasi untuk mendapatkan tranlasi angka lainnya. Lebih formalnya, misalkan diberi satu set pasangan kata dan representasi vektor yang terkait $\{x_i, z_i\}_{i=1}^{n}$, di mana $x_i \in \mathbb{R}^{d_1}$ adalah representasi terdistribusi dari kata i dalam bahasa sumber, dan $z_i \in \mathbb{R}^{d_2}$ adalah representasi vektor dari terjemahannya. Kemudian dicari matriks transformasi W sehingga Wx_i mendekati z_i. Dalam praktiknya, W dapat dipelajari dengan masalah optimasi berikut

$$\min_{W}\sum_{i=1}^{n}\|Wx_i - z_i\|^2 \quad (3)$$

yang dapat diselesaikan dengan *stochastic gradient descent* [12].

Pada saat melakukan prediksi, untuk setiap kata baru yang diberikan dan representasi vektor kontinu x, kita dapat memetakannya ke ruang bahasa lain dengan menghitung $z = Wx$. Kemudian kita dapat menemukan kata yang representasinya paling dekat dengan z dalam ruang bahasa target, menggunakan *cosine similarity* sebagai metrik jarak. Terlepas dari kesederhanaannya, metode transformasi linier ini bekerja dengan baik dalam eksperimen yang [12] jalankan, lebih baik daripada teknik *nearest neighbour* dan juga pengklasifikasi *neural network*.

Meski *monolingual mapping* sukses mendapatkan ruang *embedding* antar bahasa, teknik ini mahal dan susah diaplikasikan ke bahasa yang memiliki sumber daya rendah. Untuk dapat mengaplikasikan teknik ini diperlukan kamus kata-kata atau kalimat paralel antar bahasa, hal yang seringkali tidak tersedia pada bahasa bersumber daya rendah. Oleh karena itu, mengaplikasikan teknik ini sangat sulit pada bahasa Indonesia.

III. Transformer

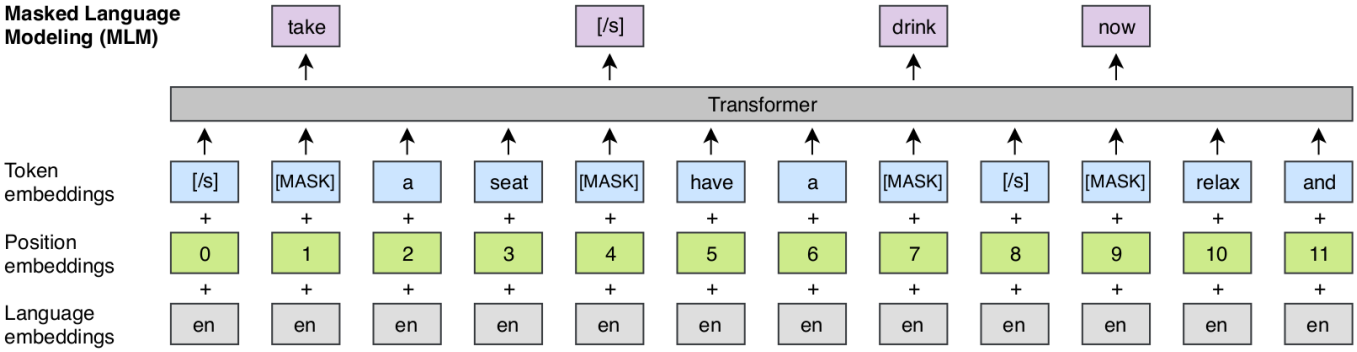
Teknik pembelajaran *Transformer*, yang termasuk jenis ***discriminative***, pertama kali dimodelkan pada tahun 2017 oleh Vaswani dkk. [14]. Perkembangan topologi ini berhasil mendorong performa pada berbagai dataset tolak ukur seperti GLUE [15]. Sub-bab selanjutnya akan menjelaskan lebih detil mengenai teknik dan penggunaannya.

A. Teknik Pembelajaran

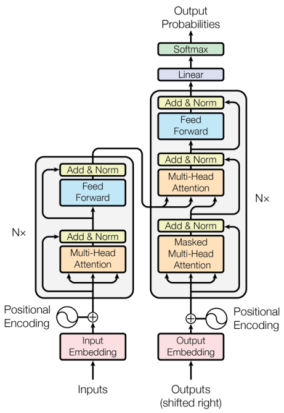
Berdasarkan [14], arsitektur *Transformer* terdiri dari *self-attention* dan *point-wise* yang ditumpuk, dan *fully connected layer* untuk enkoder dan dekoder. Ilustrasi secara keseluruhan arsitektur *Transformer* dapat dilihat pada Gambar 7.

Fungsi *attention* dapat digambarkan sebagai memetakan kueri dan pasangan *key-value* untuk suatu output, di mana kueri (Q), kunci (K), nilai (V), dan *output* semuanya vektor. *Output* dihitung sebagai *weighted sum* dari nilai-nilai, di mana bobot yang ditetapkan untuk setiap nilai dihitung oleh fungsi kompatibilitas kueri dari kunci yang sesuai.

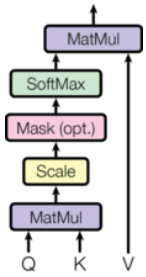
Pada penelitian [14], tipe *attention* ini disebut "*Scaled Dot-Product Attention*". Input terdiri dari kueri dan kunci dimensi d_k, dan nilai dimensi d_v. *Dot product* dari kueri dihitung dengan semua kunci, membaginya dengan $\sqrt{d_k}$, dan menerapkan fungsi softmax untuk mendapatkan bobot pada nilai. Ilustrasi *attention* dapat dilihat pada Gambar 8.



Gambar 6. Ilustrasi *masked language modeling* [2].



Gambar 7. Ilustrasi *transformer* secara keseluruhan [14].



Gambar 8. Ilustrasi *attention* secara keseluruhan [14].

Pada prakteknya, hasil keluaran fungsi *attention* pada sebuah set kueri dihitung secara besamaan, dikemas bersama menjadi matriks Q. Kunci dan nilai juga dikemas bersama menjadi matriks K dna V. Kemudian hasil keluaran dapat dihitung dengan:

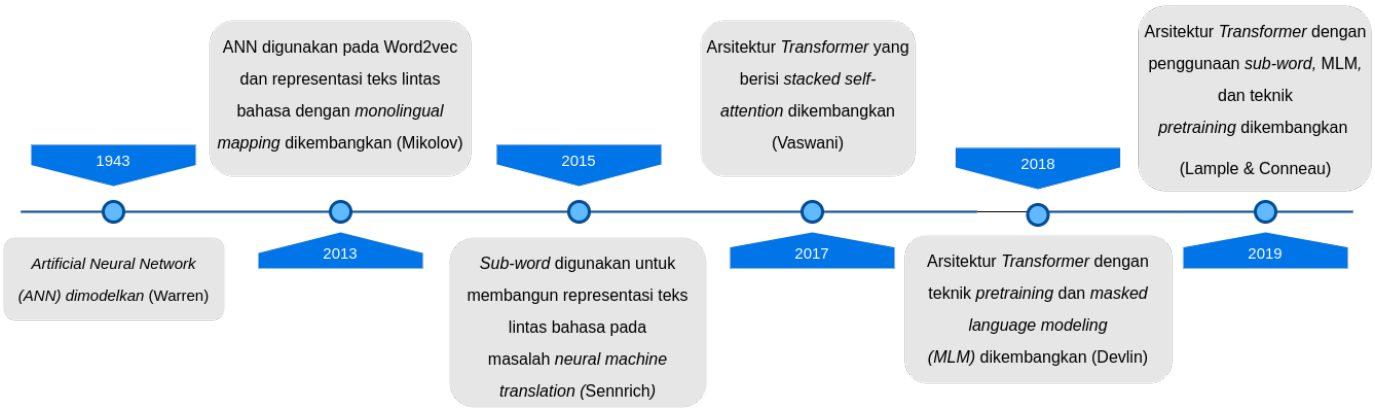
$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (4)$$

Dengan menggunakan arsitektur ini, didapatkan kemajuan terbaru dalam representasi teks dengan teknik *joint optimization*. Teknik *joint optimization* yang sebelumnya memerlukan korpus paralel atau kamus bilingual (*supervised*) [13], kini dapat dilakukan tanpa korpus paralel atau kamus bilingual (*unsupervised*) [2]. Hal ini dilakukan dengan memanfaatkan *shared sub-word vocabulary*. Tidak hanya itu, [2] juga memanfaatkan *masked language modeling* dan arsitektur *transformer* untuk mendapatkan *language model* yang mangkus. Sub-bab selanjutnya akan membahas detail *shared sub-word vocabulary* dan *masked language modeling* dari *language model pretraining* lintas bahasa bernama XLM yang dikembangkan oleh [2].

B. Shared sub-word vocabulary

Berdasarkan penelitian oleh [16], salah satu inspirasi menggunakan *sub-word* adalah fakta bahwa terjemahan beberapa kata bersifat transparan karena terjemahannya dapat diterjemahkan oleh penerjemah yang kompeten, bahkan jika itu adalah kata yang belum pernah ditemui, berdasarkan terjemahan dari *sub-word* yang dikenal seperti morfemnya atau fonemnya. Kategori kata yang terjemahannya berpotensi transparan meliputi:

1) Entitas bernama (*named entities*). Di antara bahasa yang menggunakan alfabet, nama seringkali dapat disalin dari sumber ke teks target. Transkripsi atau transliterasi mungkin diperlukan pada bahasa dengan huruf atau suku kata berbeda. Contoh:
   Barack Obama (Bahasa Inggris dan Jerman)
   Барак Обама (Bahasa Rusia)
   バラク・オバマ (ba-ra-ku o-ba-ma) (Bahasa Jepang)
2) Kata-kata serumpun dan pinjaman. Kata serumpun dan kata pinjaman dengan asal yang sama dapat berbeda secara reguler antar bahasa, sehingga aturan penerjemahan tingkat karakter sudah cukup [17]. Contoh:
   claustrophobia (Bahasa Inggris)
   Klaustrophobie (Bahasa Jerman)
   Клаустрофобия (Klaustrofobiâ) (Bahasa Rusia)
3) Kata-kata yang rumit secara morfologis. Kata-kata yang mengandung banyak morfem, misalnya yang dibentuk melalui peracikan, afiksasi, atau infleksi, dapat diterjemahkan dengan menerjemahkan morfem secara terpisah. Contoh:
   solar system (Bahasa Inggris)
   Sonnensystem (Sonne + System) (Bahasa Jerman)
   Naprendszer (Nap + Rendszer) (Bahasa Hongaria)



Gambar 9. Timeline perkembangan

Penelitian [16] membuktikan bahwa segmentasi kata-kata langka ke dalam unit *sub-word* yang sesuai sudah cukup untuk memungkinkan *neural translation network* mempelajari terjemahan yang transparan, dan menggeneralisasikan pengetahuan ini untuk menerjemahkan dan menghasilkan kata-kata yang tidak ditemui sebelumnya. Pada penelitiannya, mereka menggunakan teknik *Byte Pair Encoding* untuk mendapatkan tidak hanya *sub-word*-nya tetapi juga kompresi dari kamus-kamus kata yang ada.

Teknik *Byte Pair Encoding* (BPE) [18] adalah teknik kompresi data sederhana yang secara iteratif menggantikan pasangan *byte* yang paling sering muncu; secara berurutan dengan *byte* tunggal yang tidak digunakan. Pada penelitiannya, [16] mengadaptasi algoritma *Byte Pair Encoding* untuk segmentasi kata. Alih-alih sering menggabungkan pasangan *byte*, mereka menggabungkan karakter atau urutan karakter.

Pertama-tama, kosakata simbol diinisialisasi dengan kosakata karakter, dan mewakili setiap kata sebagai urutan karakter, ditambah simbol akhir kata khusus '·', yang memungkinkan hasil terjemahan dikembalikan ke bentuk awal setelah terjemahan. Kemudian semua pasangan simbol dihitung secara iteratif dan untuk setiap pasangan yang paling sering diganti dengan simbol baru. Contoh ('A', 'B') menjadi 'AB'. Setiap operasi penggabungan menghasilkan simbol baru yang mewakili *n-gram* dari karakter. Karakter yang sering muncul (atau seluruh kata) pada akhirnya digabungkan menjadi satu simbol. Ukuran kosa kata simbol akhir sama dengan ukuran kosa kata awal, ditambah jumlah operasi penggabungan — yang merupakan satu-satunya *hyperparameter* dari algoritma.

Dapat dilihat contoh sederhana algoritma *Byte Pair Encoding* pada Lampiran A oleh [16]. Pada algoritma ini, operasi *Byte Pair Encoding* belajar dari kamus kata 'low', 'lower', 'newest', 'widest'. Pada akhir iterasi, akan diperoleh representasi dari kamus kata dalam bentuk simbol-simbol yang dipelajari. Pada contoh tersebut, kata 'lowest' yang berada diluar kamus kata akan direpresentasikan menjadi gabungan dari simbol 'low' dan 'est'. Di sini dapat dilihat bagaimana *Byte Pair Encoding* dapat membantu merepresentasikan kata yang berada diluar kamus kata.

C. Masked Language Modeling (MLM)

Dideskripsikan pada penelitian oleh [19], *Masked Language Modeling* terinspirasi dari *Cloze task* [20]. Teknik MLM secara acak akan menyembunyikan beberapa kata dari input, dan model memiliki objektif untuk menebak kata asli dari kata yang disembunyikan tadi berdasarkan konteks yang ada di sekelilingnya. Tidak seperti pelatihan *language model* lainnya yang berjalan dari kiri-ke-kanan, pelatihan dengan objektif MLM memungkinkan representasi dari kiri dan kanan mengambil peran. Hal ini memungkinkan kita untuk melatih *Transformer* secara dua arah. Ilustrasi dari *masked language modeling* secara keseluruhan dapat dilihat pada Gambar 6.

Sama dengan BERT [19], persentase dari kata yang akan dipilih untuk disembunyikan pada XLM adalah 15 persen. Setelah sebuah posisi dipilih secara acak, sebuah kata kemudian memiliki 80 persen kemungkinan untuk disembunyikan, 10 persen kemungkinan untuk diganti menjadi sebuah kata acak, dan 10 persen kemungkinan tidak diganti sama sekali.

IV. Analisis dan Kesimpulan

Teknik representasi teks lintas bahasa telah berkembang jauh seperti yang dapat dilihat pada Gambar 9. Pada awalnya, representasi teks lintas bahasa harus didapatkan melalui melatih ANN dengan teknik CBOW atau Skip-gram. Setelah didapatkan representasi tiap bahasa, dilakukan *monolingual mapping* dengan menggunakan korpus paralel atau kamus bilingual untuk mendapatkan representasi teks lintas bahasa. Teknik ini sangat mahal untuk dilakukan dikarenakan jarang tersedia korpus paralel atau kamus bilingual untuk bahasa seperti bahasa Indonesia. Selain itu, teknik ini juga tidak mangkus dalam prakteknya dikarenakan dangkalnya model yang digunakan.

Perkembangan terbaru dengan memanfaatkan *shared sub-word vocabulary*, arsitektur *transformer*, dan teknik *masked language modeling* berhasil mempermudah dan meningkatkan efektivitas teknik sebelumnya. Dengan menggunakan *shared sub-word vocabulary*, representasi teks lintas bahasa bisa didapatkan tanpa memerlukan korpus paralel atau kamus bilingual. Selain itu, dengan menggunakan arsitektur *transformer* dan

teknik *masked language modeling*, model dapat lebih mangkus mempelajari representasi teks lintas bahasa. Hal ini dibuktikan dengan memperoleh hasil tertinggi dalam dataset tolak ukur lintas bahasa XNLI [21].

PUSTAKA

[1] Tomas Mikolov et al. "Distributed Representations of Words and Phrases andtheir Compositionality". In: arXiv:1310.4546 [cs, stat] (2013). arXiv: 1310.4546.URL :http://arxiv.org/abs/1310.4546.

[2] Guillaume Lample & Alexis Conneau. "Cross-lingual Language Model Pre-training". In: arXiv:1901.07291 [cs] (2019). arXiv: 1901.07291. URL :http://arxiv.org/abs/1901.07291.

[3] McCulloch, Warren; Walter Pitts. (1943). "A Logical Calculus of Ideas Immanent in Nervous Activity". Bulletin of Mathematical Biophysics. 5 (4): 115–133. doi:10.1007/BF02478259

[4] Ng, Andrew; Dean, Jeff. (2012). "Building High-level Features Using Large Scale Unsupervised Learning". arXiv:1112.6209

[5] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[6] Masayu Leylia Khodra & Dessi Puji Lestari. (2019). "Regresi dan ANN". URL: http://stei.kuliah.itb.ac.id

[7] Sebastian Ruder et al. "Transfer Learning in Natural Language Processing". In: Proceedings of the 2019 Conference of the North American Chapter of the As- sociation for Computational Linguistics: Tutorials. 2019, pp. 15–18.

[8] Thang Luong, Hieu Pham & Christopher D. Manning. "Bilingual Word Repre- sentations with Monolingual Quality in Mind". In: Proceedings of the 1st Work- shop on Vector Space Modeling for Natural Language Processing. Association for Computational Linguistics, 2015, 151–159. URL : https://www.aclweb.org/anthology/W15-1521. DOI :10.3115/v1/W15-1521.

[9] Zirui Wang et al. "Cross-lingual Alignment vs Joint Training: A Comparative Study and A Simple Unified Framework". In: arXiv:1910.04708 [cs] (2019). arXiv: 1910.04708. URL : http://arxiv.org/abs/1910.04708.

[10] Tomas Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: arXiv:1301.3781 [cs] (2013). arXiv: 1301.3781. URL : https://arxiv.org/abs/1301.3781.

[11] Tomas Mikolov, Wen-tau Yih & Geoffrey Zweig. "Linguistic Regularities in Continuous Space Word Representations". In: Proceedings of the 2013 Con- ference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2013, 746–751. URL : https://www.aclweb.org/anthology/N13-1090.

[12] Tomas Mikolov, Quoc V. Le & Ilya Sutskever. "Exploiting Similarities among Languages for Machine Translation". In: arXiv:1309.4168 [cs] (2013). arXiv: 1309.4168. URL : http://arxiv.org/abs/1309.4168.

[13] Chao Xing et al. "Normalized Word Embedding and Orthogonal Transform for Bilingual Word Translation". In: Proceedings of the 2015 Conference of the Nor- th American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2015.

[14] Ashish Vaswani et al. "Attention Is All You Need". In: arXiv:1706.03762 [cs] (2017). arXiv: 1706.03762. URL : http://arxiv.org/abs/1706.03762.

[15] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: arXiv:1804.07461 [cs] (2019). arXiv: 1804.07461. URL : http://arxiv.org/abs/1804.07461.

[16] Rico Sennrich, Barry Haddow & Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". In: arXiv:1508.07909 [cs] (2016). arXiv: 1508.07909. URL : http://arxiv.org/abs/1508.07909.

[17] Jörg Tiedemann. "Character-Based Pivot Translation for Under-Resourced Languages and Domains". In: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2012, 141–151. URL : https://www.aclweb.org/anthology/E12-1015.

[18] Philip Gage. "A New Algorithm for Data Compression". In: C Users J. 12.2 (1994), 23–38. ISSN : 0898-9788.

[19] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: arXiv:1810.04805 [cs] (2019). arXiv: 1810.04805. URL : http://arxiv.org/abs/1810.04805.

[20] Wilson L. Taylor. ""Cloze Procedure": A New Tool for Measuring Readability". In: Journalism Quarterly 30.4 (1953), 415–433. ISSN : 0022-5533. DOI :10.1177/107769905303000401.

[21] Alexis Conneau et al. "XNLI: Evaluating Cross-lingual Sentence Representations". In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2018, 2475–2485. DOI : 10.18653/v1/D18- 1269. URL :https://www.aclweb.org/anthology/D18-1269.

LAMPIRAN A
ALGORITMA *Byte Pair Encoding* SEDERHANA

Algoritma (contoh nama file: *bpe.py*):

```
import re, collections
def get_stats(vocab):
    pairs = collections.defaultdict(int)
    for word, freq in vocab.items():
        symbols = word.split()
        for i in range(len(symbols)-1):
            pairs[symbols[i],symbols[i+1]] += freq
    return pairs

def merge_vocab(pair, v_in):
    v_out = {}
    bigram = re.escape(' '.join(pair))
    p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)'
  )
    for word in v_in:
        w_out = p.sub(''.join(pair), word)
        v_out[w_out] = v_in[word]
    return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
'n e w e s t </w>':6, 'w i d e s t </w>':3}
vocab_test = {'l o w e s t </w>': 1}

num_merges = 10
for i in range(num_merges):
    pairs = get_stats(vocab)
    best = max(pairs, key=pairs.get)
    print('~~~')
    vocab = merge_vocab(best, vocab)
    vocab_test = merge_vocab(best, vocab_test)
    print("best: ", best)
    print("vocab: ", vocab)
    print("vocab_test: ", vocab_test)

```

Setelah dijalankan di mesin bersistem operasi Ubuntu 18.04 dengan perintah

```
    $ python3 bpe.py

```

akan didapatkan keluaran sebagai berikut:

```
~~~
best:  ('e', 's')
vocab:  {'l o w </w>': 5, 'l o w e r </w>': 2, 'n
  e w es t </w>': 6, 'w i d es t </w>': 3}
vocab_test:  {'l o w es t </w>': 1}
~~~
best:  ('es', 't')
vocab:  {'l o w </w>': 5, 'l o w e r </w>': 2, 'n
  e w est </w>': 6, 'w i d est </w>': 3}
vocab_test:  {'l o w est </w>': 1}
~~~
best:  ('est', '</w>')
vocab:  {'l o w </w>': 5, 'l o w e r </w>': 2, 'n
  e w est</w>': 6, 'w i d est</w>': 3}
vocab_test:  {'l o w est</w>': 1}
~~~
best:  ('l', 'o')
vocab:  {'lo w </w>': 5, 'lo w e r </w>': 2, 'n e
  w est</w>': 6, 'w i d est</w>': 3}
vocab_test:  {'lo w est</w>': 1}
~~~
best:  ('lo', 'w')
vocab:  {'low </w>': 5, 'low e r </w>': 2, 'n e w
  est</w>': 6, 'w i d est</w>': 3}
vocab_test:  {'low est</w>': 1}
~~~
best:  ('n', 'e')
vocab:  {'low </w>': 5, 'low e r </w>': 2, 'ne w
  est</w>': 6, 'w i d est</w>': 3}
vocab_test:  {'low est</w>': 1}
~~~
best:  ('ne', 'w')
vocab:  {'low </w>': 5, 'low e r </w>': 2, 'new
  est</w>': 6, 'w i d est</w>': 3}
vocab_test:  {'low est</w>': 1}
~~~
best:  ('new', 'est</w>')
vocab:  {'low </w>': 5, 'low e r </w>': 2, 'newest
  </w>': 6, 'w i d est</w>': 3}
vocab_test:  {'low est</w>': 1}
~~~
best:  ('low', '</w>')
vocab:  {'low</w>': 5, 'low e r </w>': 2, 'newest
  </w>': 6, 'w i d est</w>': 3}
vocab_test:  {'low est</w>': 1}
~~~
best:  ('w', 'i')
vocab:  {'low</w>': 5, 'low e r </w>': 2, 'newest
  </w>': 6, 'wi d est</w>': 3}
vocab_test:  {'low est</w>': 1}

```