



**PENENTUAN KETERIKATAN TEKSTUAL SURAT
KONTRAK *NON-DISCLOSURE AGREEMENT*
MENGUNAKAN *ALBERT***

Skripsi

Disusun Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Komputer
Program Studi Teknik Informatika

Oleh

Abdillah Azmi

4611420020

**PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI SEMARANG
2024**

PERSETUJUAN PEMBIMBING

Proyek skripsi berjudul “Penentuan Keterikatan Tekstual Surat Kontrak *Non-Disclosure Agreement* Menggunakan ALBERT” yang disusun oleh

Nama : Abdillah Azmi

NIM : 4611420020

Prodi : Teknik Informatika

Telah disetujui untuk diajukan ke sidang ujian proyek skripsi.

Semarang, 2 Mei 2024

Pembimbing,

A handwritten signature in black ink, appearing to read 'Alamsyah', is placed over a light blue rectangular background.

Dr. Alamsyah, S.Si., M.Kom.

NIP 197405172006041001

PENGESAHAN PENGUJI




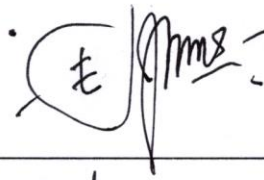


Skripsi berjudul “Penentuan Keterikatan Tekstual Dalam Surat Kontrak Bertipe NDA (*Non-Disclosure Agreement*) Menggunakan ALBERT (*A Lite BERT*)” yang disusun oleh

Nama : Abdillah Azmi

NIM : 4611420020

Prodi : Teknik Informatika

Telah dipertahankan dalam ujian sidang skripsi pada hari Kamis, 16 Mei 2024

Ketua Penguji Prof. Edy Cahyono, M.Si. NIP. 196412051990021001	  UNNES FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
Sekretaris Dr. Alamsyah, S.Si., M.Kom. NIP. 197405172006041001	
Penguji 1 Endang Sugiharti S.Si., M.Kom. NIP. 197401071999032001	
Penguji 2 Yahya Nur Ifriza S.Pd., M.Kom. NIP. 199001172022031008	
Pembimbing Dr. Alamsyah, S.Si., M.Kom. NIP. 197405172006041001	

PERNYATAAN KEASLIAN

Skripsi yang ditulis berjudul “Penentuan Keterikatan Tekstual Surat Kontrak *Non-Disclosure Agreement* Menggunakan ALBERT” merupakan karya ilmiah asli dan bukan hasil plagiasi dari karya ilmiah orang lain. Pendapat atau temuan orang lain yang dikutip di dalam skripsi ini telah ditulis berdasarkan kode etik ilmiah.

Semarang, 22 Juli 2024

Yang Menyatakan

Abdillah Azmi

NIM. 4611420020

MOTO DAN PERSEMBAHAN

Moto:

“Tidak semua harus dipikirkan, dipedulikan, dan diingat. Harus efisien dengan tenaga & pikiran”

Persembahan

Skripsi ini penulis persembahkan kepada orang tua dan saudara yang telah memotivasi peneliti untuk terus belajar dan mengejar mimpi hingga sampai di titik ini.

ABSTRAK

Azmi, Abdillah Azmi. 2024. Penentuan Keterikatan Tekstual Surat Kontrak *Non-Disclosure Agreement* Menggunakan ALBERT. Skripsi. Program Studi Teknik Informatika. Fakultas Matematika dan Ilmu Pengetahuan Alam. Universitas Negeri Semarang. Pembimbing Dr. Alamsyah, S.Si., M.Kom.

Kata Kunci: NLI, ALBERT, *Textual Entailment*, NLP, *Contract*, *Finetuning*, *NDA*, *Deep Learning*, *Language Model*.

NDA (*Non-Disclosure Agreement*) merupakan salah satu tipe surat kontrak. NDA mengikat dua atau lebih pihak dimana semuanya setuju bahwa informasi tertentu yang dibagikan atau dibuat satu pihak bersifat rahasia. Kontrak ini berfungsi untuk melindungi informasi sensitif, menjaga hak paten, dan mengontrol informasi yang dibagikan. Membaca dan memahami surat kontrak adalah proses yang repetitif, memakan banyak waktu dan tenaga. Walau begitu, aktivitas tersebut masih sangat krusial dalam dunia bisnis, karena dapat mengikat dua atau lebih pihak dibawah hukum. Masalah ini sangat cocok untuk dikerjakan oleh *Artificial Intelligence* menggunakan *Deep Learning*. Oleh karena itu, penelitian ini bertujuan untuk menguji dan mengembangkan *pretrained language model* yang didesain khusus untuk memahami surat kontrak. Metode yang digunakan adalah melatih model untuk melakukan tugas inferensi bahasa berupa *textual entailment* dengan dataset ContractNLI. Hasilnya, model ALBERT-base yang digunakan menunjukkan skor akurasi sebesar 85,0% dan skor F1 makro sebesar 82,4%. Walaupun nilai ini tidak menjadi *State of The Art* dari *benchmark* ContractNLI, model yang dilatih masih bisa mengalahkan beberapa model lain yang memiliki parameter jauh lebih besar.

PRAKATA

Puji Syukur kehadiran Allah SWT yang telah melimpahkan rahmat dan nikmat. Shalawat serta salam tak lupa kita haturkan kepada Nabi Muhammad SAW. Semoga kita mendapatkan syafaat Beliau di hari akhir nanti. Alhamdulillah, dengan penuh rasa syukur penulis mengungkapkan bahwa skripsi dengan judul “Penentuan Keterikatan Tekstual Surat Kontrak *Non-Disclosure Agreement* Menggunakan ALBERT” dapat terselesaikan dengan lancar. Penulis mengucapkan rasa terima kasih kepada pihak-pihak yang telah membantu dan mendukung dalam penyusunan skripsi, terutama kepada.

1. Prof. Dr. S Martono, M.Si, selaku Rektor Universitas Negeri Semarang, atas kepemimpinannya yang cemerlang.
2. Prof. Dr. Edy Cahyono, M.Si. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
3. Dr. Alamsyah, S.Si., M.Kom., selaku Koordinator Prodi Sarjana Teknik Informatika Fakultas Matematika dan Ilmu Pengetahuan Alam dan selaku dosen pembimbing skripsi yang telah memberikan dukungan, masukan, dan saran kepada penulis.
4. Subhan, S.P., M.Pd., M.Kom. selaku dosen wali yang telah membimbing penulis selama menempuh masa studi.
5. Endang Sugiharti, S.Si., M.Kom., selaku Penguji 1 dan Yahya Nur Ifriza S.Pd., M.Kom., selaku Penguji 2, yang telah memberikan kritik dan saran yang membangun dalam penelitian ini.
6. Ibu dan Ayah yang telah memberi dukungan baik secara keungan maupun mental penulis sampai sejauh ini, dari masa awal kuliah hingga selesai dalam mengerjakan skripsi.
7. Teman-teman dari Teknik Informatika 2020 yang telah membagi informasi dan saran dalam menyusun skripsi.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan karena adanya keterbatasan pengetahuan dan kemampuan. Oleh karena itu, dengan segala kerendahan hati penulis memohon maaf atas segala kekurangan. Penulis juga

mengharapkan segala bentuk saran dan kritik yang membangun dari berbagai pihak sebagai bentuk pembelajaran di masa mendatang. Terakhir, penulis juga berharap semoga skripsi ini dapat memberikan manfaat diberbagai kalangan.

Semarang. 18 Mei 2024

Yang menyatakan

A handwritten signature in black ink, consisting of a large, stylized capital 'A' followed by a series of loops and a horizontal stroke at the end.

Abdillah Azmi

NIM. 4611420020

DAFTAR ISI

HALAMAN SAMPUL.....	i
PERSETUJUAN PEMBIMBING.....	ii
PENGESAHAN PENGUJI.....	iii
PERNYATAAN KEASLIAN.....	iv
MOTO DAN PERSEMBAHAN	v
ABSTRAK.....	vi
PRAKATA.....	vii
DAFTAR ISI.....	ix
DAFTAR TABEL.....	xi
DAFTAR GAMBAR	xii
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	4
1.3 Batasan Masalah.....	4
1.4 Tujuan Penelitian	5
1.5 Manfaat Penelitian	5
1.6 Keaslian Penelitian.....	6
BAB 2 KAJIAN PUSTAKA.....	10
2.1 Tinjauan Pustaka	10
2.2 Landasan Teori.....	12
2.2.1 <i>Natural Language Processing (NLP)</i>	12
2.2.2 Transformer.....	16
2.2.3 <i>Natural Language Inference (NLI)</i>	19
2.2.4 <i>Textual Entailment</i>	22
2.2.5 <i>Surat Kontrak</i>	24
2.2.6 <i>NDA (Non-Disclosure Agreement)</i>	26
2.2.7 <i>EDGAR</i>	27
2.2.8 <i>BERT</i>	28
2.2.9 <i>ALBERT</i>	29
2.2.10 <i>PyTorch</i>	30
BAB 3 METODE PENELITIAN.....	32

3.1	Pendekatan dan Desain Penelitian	32
3.2	Lokasi Penelitian.....	35
3.3	Fokus Penelitian	35
3.4	Data dan Sumber Data	35
3.5	Teknik Pengumpulan Data.....	38
3.6	Teknik Keabsahan Data	39
3.7	Teknik Analisis Data.....	40
3.7.1	Analisis Metode	40
3.7.2	Analisis Hasil	40
BAB 4 HASIL PENELITIAN DAN PEMBAHASAN		42
4.1	Hasil Penelitian	42
4.1.1	Hasil Pengolahan Data	42
4.1.2	Hasil Implementasi ALBERT	49
4.1.3	Hasil Implementasi Aplikasi Web	60
4.2	Pembahasan.....	65
BAB 5 PENUTUP		70
5.1	Kesimpulan	70
5.2	Saran.....	71
DAFTAR PUSTAKA		72
LAMPIRAN.....		79
Lampiran 1. Biodata Penulis		79
Lampiran 2. SK Pembimbing		80
Lampiran 3. SK Penguji		81
Lampiran 4. Source Code		82

DAFTAR TABEL

Table 1. 1 Penelitian terdahulu menggunakan model ALBERT.....	6
Tabel 2. 1 Sampel kasus dalam dataset S NLI (<i>Stanford Natural Language Inference</i>).....	23
Tabel 2. 2 Perbandingan model BERT dan ALBERT	29
Tabel 3. 1 Pemecahan data berdasarkan sumber dan format dokumen kontrak ...	37
Tabel 3. 2 Persebaran kelas berdasarkan hipotesis. (Judul hipotesis hanya bertujuan supaya mudah dibaca).....	37
Tabel 3. 3 Struktur tabel dataset dari ContractNLI	38
Tabel 3. 4 Contoh <i>Hyperparamter</i> untuk model ALBERT	40
Tabel 4. 1 Detail hasil tokenisasi premis dan hipotesis dataset CNLI.....	48
Tabel 4. 2 <i>Hyperparameter</i> yang dipakai dalam proses training	52
Tabel 4. 3 Layer model ALBERT-base dengan konfigurasi 1 (satu) layer group dan classifier head	53
Tabel 4. 4 Hasil eksperimen <i>pre-training</i> model dengan satu hidden layer group	57
Tabel 4. 5 Hasil eksperimen pre-training model dengan tiga <i>hidden layer group</i>	58
Tabel 4. 6 Hasil utama eksperimen dengan nilai akurasi terbaik.....	66
Tabel 4. 7 Hasil EM (<i>Exact Match</i>) dari model	67

DAFTAR GAMBAR

Gambar 2. 1 Perkembangan terkini dalam NLP	13
Gambar 2. 2 <i>Word Embedding</i> dengan vektor 4 dimensi	14
Gambar 2. 3 Arsitektur encoder-decoder Transformer	17
Gambar 2. 4 Tiga area penelitian NLI dalam komunitas <i>Natural Language Processing</i>	19
Gambar 2. 5 Contoh Surat Pernjajian Kerahasiaan (NDA)	27
Gambar 2. 6 Sistem pencarian pada sistem EDGAR	28
Gambar 3. 1 Alur kerja proses eksperimen penelitian	33
Gambar 3. 2 Arsitektur model BERT dan ALBERT	34
Gambar 3. 3 Struktur file JSON dari CNLI sebelum dilakukan pre-processing... ..	36
Gambar 4. 1 Grafik distribusi kata dari teks surat kontrak CNLI (premis)	43
Gambar 4. 2 Grafik distribusi kata dalam hipotesis dari dataset CNLI	44
Gambar 4. 3 Distribusi data CNLI berdasarkan label <i>entail</i> , <i>contradict</i> , dan <i>neutral</i>	44
Gambar 4. 4 Grafik persebaran label (kelas) dalam tiap hipotesis	46
Gambar 4. 5 Grafik persebaran data pada subset <i>training</i> , <i>validation</i> (dev), dan <i>testing</i>	47
Gambar 4. 6 Grafik distribusi panjang token pada tiap teks surat kontrak (premis) dataset CNLI	48
Gambar 4. 7 Alur proses pre-training model dengan data CNLI	50
Gambar 4. 8 Detail alur proses training model ALBERT-base	51
Gambar 4. 9 Source code penggunaan model ALBERT-base dari huggingface.co	53
Gambar 4. 10 <i>Source code</i> penggunaan optimizer AdamW	55
Gambar 4. 11 <i>Source code</i> penggunaan early stopping dengan lambdaLR pada optimizer	56
Gambar 4. 12 Hasil <i>Unnormalized multi-class confusion matrix</i> dari subset data testing	59
Gambar 4. 13 Halaman Home dari aplikasi web	60
Gambar 4. 14 Halaman Abstrak dari aplikasi Web	61

Gambar 4. 15 Halaman Referensi dari penelitian	62
Gambar 4. 16 Halaman gambar dari penelitian	63
Gambar 4. 17 Halaman tabel dari penelitian.....	63
Gambar 4. 18 Halaman download file skripsi.....	64
Gambar 4. 19 Halaman About me dari aplikasi web	64
Gambar 4. 20 Halaman demo dari aplikasi.....	65

BAB 1

PENDAHULUAN

1.1 Latar Belakang

NLP (*Natural Language Processing*) adalah teori yang mempelajari hubungan antara komputer dan bahasa manusia melalui teknik kecerdasan buatan (Khan dkk., 2023). NLP memungkinkan komputer untuk mengerjakan tugas-tugas yang melibatkan bahasa, seperti penerjemahan, tanya-jawab, atau klasifikasi teks. Seiring berkembangnya teknologi komputasi, pendekatan terhadap NLP cenderung mengarah ke *Deep Learning*, atau arsitektur dengan lapisan layer yang sangat banyak. Sebelum adanya *Deep Learning*, mayoritas tugas NLP dikerjakan dengan pendekatan generatif atau diskriminatif seperti HMM (*Hidden Markov Model*), CRF (*Conditional Random Fields*), dan SVM (*Support Vector Machine*). Saat ini, pendekatan yang paling populer dalam area NLP adalah menggunakan arsitektur RNN (*Recurrent Neural Network*), CNN (*Convolutional Neural Network*), atau Transformer.

Transformer (Vaswani dkk., 2017) merupakan arsitektur yang paling dominan dalam NLP, melebihi alternatif lain seperti CNN dan RNN dalam hal performa baik dalam *natural language understanding* maupun *natural language generation*. Kemampuan arsitektur ini meningkat seiring dengan *training* data dan ukuran model, memungkinkan untuk melakukan *parallel training* dan menangkap sekuen fitur teks secara efisien (Wolf dkk., 2020). *Pretrained* model adalah model NLP yang telah dilatih dengan korpus besar dan dapat disesuaikan dengan tugas spesifik dengan performa yang bagus. Tugas-tugas tersebut dapat berupa klasifikasi, NER (*Named Entity Recognition*), penerjemah, tanya jawab, atau PoS (*Part of Speech*) tagging (Khan dkk., 2023). Arsitektur Transformer sangat cocok untuk melakukan pretraining dengan korpus teks yang sangat besar, menghasilkan akurasi yang bagus pada tugas-tugas khusus misalnya klasifikasi teks. Salah satu *pretrained* model yang ada adalah BERT (Devlin dkk., 2019).

BERT (*Bidirectional Encoder Representations from Transformer*) adalah pretrained bidirectional model yang didesain untuk memahami konteks dalam teks dari dua arah pada tiap layer (Devlin dkk., 2019). BERT memanfaatkan dua metode dalam proses pretraining untuk melakukan bidirectional training, yaitu MLM (*Masked Language Model*) dan NSP (*Next Sentence Prediction*). MLM memiliki tujuan untuk melatih model untuk memahami teks dengan cara melengkapi kata yang hilang dari suatu sekuen teks. Sedangkan NSP menuntut model untuk belajar dengan cara memprediksi kalimat lanjutan dari suatu teks. BERT menyediakan dua tipe model, yaitu BERT-base dengan 110 juta parameter dan BERT-large dengan 340 juta parameter. Saat ini, sudah banyak versi model yang lahir dari model BERT. Beberapa model tersebut adalah DistilBERT (Sanh dkk., 2019), RoBERTa (Liu dkk., 2019), DeBERTa (He dkk., 2020), dan ALBERT (Lan dkk., 2020).

ALBERT (*A Lite BERT*) adalah model yang lahir sebagai versi peningkatan dari model pendahulunya, BERT (Devlin dkk., 2019). Dengan fokus pada pengurangan ukuran dan kompleksitas layer dengan tetap menjaga efisiensi dan performa, ALBERT memanfaatkan 2 (dua) metode sebagai peningkatan dari model pendahulunya. Pertama adalah *cross-layer parameter sharing*, yang memungkinkan model untuk berbagi (*reuse*) parameter yang sama. Terdapat tiga cara bagi model untuk menggunakan *parameter sharing*, yaitu dengan berbagi parameter *feed forward network*, *parameter attention head*, dan berbagi semua parameter. Kedua adalah SOP (*Sentence Order Prediction*) loss, yang menuntut model untuk belajar dengan memahami dua kalimat dan urutannya dalam teks, tidak hanya struktur katanya saja. SOP loss sangat cocok untuk memahami sekuen teks panjang dan melakukan tugas inferensi atau pemahaman bahasa.

Natural Language Inference (NLI) merujuk pada kemampuan mesin dalam memahami bahasa secara mendalam dan tak hanya apa yang diungkapkan secara eksplisit, melainkan mengandalkan kesimpulan yang diambil dari pengetahuan tentang bagaimana dunia bekerja (Storks dkk., 2020). Pengetahuan yang dibutuhkan bagi komputer untuk memiliki performa NLI yang baik adalah pengetahuan linguistik, pengetahuan umum, dan penalaran. Untuk menguji

performa NLI tersebut, terdapat berbagai benchmark yang menyediakan bermacam-macam tugas yang telah didesain khusus untuk model NLI, seperti tugas tanya-jawab atau keterikatan tekstual (*textual entailment*). Tugas-tugas ini sangat penting bagi model atau algoritma NLI untuk menentukan seberapa baik komputer dalam memahami teks atau dokumen dalam bahasa atau domain tertentu. Contohnya adalah dokumen surat kontrak perjanjian kerahasiaan atau NDA (*Non-Disclosure Agreement*).

Membaca dan memahami dokumen-dokumen kontrak merupakan proses yang memakan banyak waktu dan tenaga. 60 sampai 80% transaksi bisnis-ke-bisnis dilakukan dengan semacam persetujuan tertulis, dengan umumnya 1000 perusahaan menangani 20.000 sampai 40.000 kontrak tiap waktunya (Exigent Group Limited, 2019). Meninjau kontrak umumnya dilakukan manual oleh profesional, dengan biaya yang tidak sedikit terutama untuk perusahaan yang menangani banyak dokumen tiap tahunnya. Proses ini sangat membebani organisasi atau perusahaan skala kecil. Untuk menangani masalah ini, tumbuh area dalam *Artificial Intelligence* terutama NLP yang berfokus pada memahami dokumen kontrak secara otomatis.

CNLI (*Contract Natural Language Inference*) (Koreeda & Manning, 2021) adalah benchmark yang dirancang untuk menguji kemampuan model dalam tugas inferensi bahasa dalam domain hukum dengan dataset berupa dokumen-dokumen surat kontrak NDA yang memiliki rata-rata token per-dokumen sebesar 2.254. CNLI mengandung total 607 dokumen, dimana tiap dokumen terdiri lebih dari 2 halaman. Selain itu, dataset juga mengandung 17 hipotesis yang dibuat secara manual untuk memastikan tingkat kesulitan yang tidak mudah. *Benchmark* ini dapat menguji suatu model dalam dua tugas, yaitu NLI dengan tipe *textual entailment* dan *evidence identification*. Total pasangan dokumen dan hipotesis dari dataset CNLI adalah 10.319 sampel, dan dibagi menjadi train, dev, test, sebesar 70%, 10%, 20%. Baseline model yang disediakan oleh (Koreeda & Manning, 2021) adalah SpanNLI BERT-base dan SpanNLI BERT-large, yang menghasilkan skor akurasi sebesar 83,8 dan 87,5.

SCROLLS (Shaham dkk., 2022) merupakan metode yang didesain untuk menyelesaikan tugas yang membutuhkan pemahaman teks panjang. SCROLLS

menggunakan basis model BART (Lewis dkk., 2019) dan LED (*Longformer Encoder-Decoder*). Performa model ini dalam benchmark CNLI adalah skor EM (*Exact Match*) sebesar 77,4 pada versi BART dan 73,4 pada versi LED. Lalu terdapat CoLT5 (*Conditional Long T5*) (Ainslie dkk., 2023) yang memanfaatkan basis model T5 yang dimodifikasi menangani input panjang. Model ini menghasilkan skor EM sebesar 88,7 pada benchmark CNLI. Selain itu, terdapat model SLED (*Sliding Encoder and Decoder*) yang juga didesain untuk memproses input teks panjang. Basis model yang digunakan adalah BART, T5, dan LED. Hasilnya, SLED dengan basis BART menghasilkan skor EM tertinggi dibanding model SLED lain, yaitu sebesar 87,3.

Berdasarkan latar belakang yang telah diuraikan, peneliti memutuskan untuk melakukan eksperimen untuk mendesain model berbasis ALBERT yang berfokus pada tugas inferensi keterikatan tekstual khusus dalam domain hukum. Lalu model akan diuji dengan benchmark CNLI untuk mengevaluasi performanya dan peningkatan dibanding model lain yang sudah ada. Tujuannya adalah berkontribusi dalam pengembangan deep learning dalam ranah hukum. Maka dari itu peneliti mengajukan judul penelitian **“Penentuan Keterikatan Tekstual Dalam Surat Kontrak Bertipe NDA (*Non-Disclosure Agreement*) Menggunakan ALBERT (*A Lite BERT*)”**

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, masalah dalam penelitian dapat dirumuskan sebagai berikut.

1. Bagaimana cara melakukan *finetuning* dan *pretraining* model ALBERT-base untuk melakukan inferensi keterikatan tekstual dengan data surat kontrak NDA (*Non-Disclosure Agreement*)?
2. Bagaimana performa model ALBERT-base dalam melakukan inferensi keterikatan tekstual dengan dataset CNLI (*Contract NLI*)?

1.3 Batasan Masalah

Untuk memastikan penelitian terfokus pada permasalahan dan berjalan dengan yang telah direncanakan, perlu didefinisikan batasan-batasan penelitian. Adapun batasan yang dimiliki oleh penelitian ini adalah sebagai berikut.

1. Data yang digunakan adalah *ContractNLI* (CNLI) yang didesain khusus untuk menguji kemampuan NLI dari berbagai model dan dibuat oleh (Koreeda & Manning, 2021).
2. Data yang digunakan dalam proses *training* adalah data yang telah dibersihkan dengan menghilangkan karakter *non-alphabetical*, kata-kata yang tidak berkontribusi pada makna teks, dan mengubah semua huruf kapital menjadi kecil.
3. Dokumen yang digunakan untuk data adalah *Non-Disclosure Agreement* (NDA) atau surat perjanjian kerahasiaan berbahasa Inggris.
4. Model yang dipilih untuk menjalankan penelitian adalah ALBERT (*A Little BERT*) oleh (Lan dkk., 2020).
5. Tipe tugas NLI yang dianalisis adalah *textual entailment* atau keterkaitan tekstual antara premis dan hipotesis.
6. Model yang dilatih hanya bekerja pada data CNLI dan sangat tidak direkomendasikan untuk diimplementasikan pada ranah hukum secara langsung.

1.4 Tujuan Penelitian

1. Mengetahui cara melakukan *finetuning* dan *pretraining* model ALBERT-base untuk melakukan inferensi keterikatan tekstual dengan data surat kontrak NDA (*Non-Disclosure Agreement*).
2. Mengetahui performa model ALBERT-base dalam melakukan inferensi keterikatan tekstual dengan dataset CNLI (*Contract NLI*).

1.5 Manfaat Penelitian

1. Menunjukkan bahwa model ALBERT-base dapat memiliki performa bagus dalam tugas inferensi bahasa dengan data surat kontrak NDA.

2. Memberikan analisa hasil penerapan model ALBERT-base pada tugas inferensi bahasa dengan data surat kontrak NDA.
3. Menunjukkan kelebihan dari model ALBERT-base dalam melakukan tugas inferensi bahasa dibandingkan *language model* lain.

1.6 Keaslian Penelitian

Beberapa penelitian tentang NLP yang telah dilakukan dengan menggunakan model ALBERT dapat dilihat dalam Tabel 1.1. Setiap penelitian, baik menggunakan metode pre-training atau tidak, berusaha memberikan akurasi terbaik dengan mengutamakan efisiensi waktu dan memori dengan ukuran model yang relatif kecil.

Table 1. 1 Penelitian terdahulu menggunakan model ALBERT

Peneliti	Judul Penelitian	Metode
(Servan dkk., 2024)	<i>mALBERT: Is a Compact Multilingual BERT Model Still Worth It?</i>	<i>Pretraining</i> dan <i>Finetuning</i> ALBERT dengan data multi-bahasa dari Wikipedia.
(Balagansky & Gavrilov, 2022)	<i>PALBERT: Teaching ALBERT to Ponder</i>	Melatih model ALBERT dengan versi PonderNet yang ditingkatkan dengan <i>deterministic Q-exit criterion</i> untuk melakukan <i>early exit</i>
(Fu dkk., 2023)	<i>Energy-efficient Task Adaptation for NLP Edge Inference Leveraging Heterogeneous Memory Architectures</i>	Versi <i>multi-task</i> ALBERT yang dioptimasi untuk berfokus pada <i>data reusable</i> dan efisiensi, bernama adapter-ALBERT.
(Naseem dkk., 2021)	<i>BioALBERT: A Simple and Effective Pre-trained</i>	<i>Pretraining</i> ALBERT dengan 4,5 miliar kata dari

	<i>Language Model for Biomedical Named Entity Recognition</i>	PubMed dan 13,5 miliar kata dari PubMed Central. <i>Finetuning</i> untuk tugas BioNER (<i>Bio Named Entity Recognition</i>)
(Cañete dkk., 2022)	<i>ALBETO and DistilBETO: Lightweight Spanish Language Models</i>	<i>Pretraining</i> dan <i>Finetuning</i> ALBERT dengan data corpus berbahasa Spanyol yang berjumlah sekitar 3 miliar kata

Servan dkk., (2024) mengenalkan mALBERT, yang merupakan model ALBERT dengan kemampuan multi-bahasa. mALBERT menggunakan metode pretraining dan corpus dari kumpulan data Wikipedia. Corpus ini memiliki sekitar 21 miliar kata yang terdiri dari 50 bahasa yang paling sering digunakan di Wikipedia, termasuk bahasa Inggris dan Basque. Pretraining dilakukan dengan hyperparameter milik original ALBERT, yakni 128 batch size dan learning rate $3,125 \times 10^{-4}$. mALBERT menyediakan 3 model dengan ukuran input vocabulary berbeda, yaitu mALBERT-128k, mALBERT-64k, dan mALBERT-32k. Hasilnya, dalam benchmark seperti MMNLU dan MultiATIS++, model ini tidak melebihi performa model multi-bahasa lain seperti mBERT dan Distil-mBERT, namun masih dapat berkompetisi dengan baik.

Balagansky & Gavrilov, (2022) mengenalkan PALBERT, metode baru untuk melakukan early exit yang diaplikasikan dengan model ALBERT. Early exit dilakukan dengan tujuan untuk mempersingkat proses inferensi (pemahaman) pada sampel data yang tidak terlalu kompleks. Early exit dilakukan dengan menggunakan PonderNet. Metode baru yang digunakan bernama Q-exit (Quantile-exit). Hasilnya, setelah dievaluasi dengan benchmark GLUE (*General Language Understanding Benchmark*), PALBERT memiliki performa lebih tinggi dari model ALBERT yang tidak menggunakan Q-exit.

(Fu dkk., 2023) mengembangkan adapter-ALBERT, versi dari model ALBERT yang didesain untuk menangani multi-tasking dan fokus pada data reusable dan efisiensi penggunaan hardware. Supaya model memiliki kemampuan untuk melakukan berbagai tugas inferensi seperti parafrase, analisis sentimen, dan tanya jawab, umumnya model perlu melalui proses fine-tuning. Namun, dengan *fine-tuning*, model dapat melupakan informasi yang telah didapat dari proses tuning sebelumnya. Maka dari itu, adapter-ALBERT memanfaatkan residual adapter. Hasilnya adapter-ALBERT memiliki mayoritas parameter dari *vanilla* ALBERT ditambah *task-specific-layer* yang dapat dilatih untuk tugas inferensi yang diinginkan.

(Naseem dkk., 2021) mengenalkan BioALBERT, versi dari model ALBERT yang didesain untuk tugas NER (*Named Entity Recognition*) yang bersifat domain-specific. BioALBERT dilatih melalui proses pretraining dengan data yang berasal dari PubMed dan PMC (PubMed Central). Setelah itu, model diimplementasikan finetuning dengan dataset NER yang dibagi menjadi 4 (empat) kelas atau entitas. Entitas disease menggunakan dataset NCBI dan BC5CDR, entitas spesies menggunakan dataset SPECIES-800 dan LINNEAEUS, entitas drug/protein menggunakan dataset BC2GN dan JNLPBA, lalu entitas drug/chem menggunakan dataset BC5CDR dan BC4CHMED.

(Cañete dkk., 2022) mengenalkan model bernama ALBETO, yang merupakan versi dari model ALBERT dan didesain khusus untuk bahasa Spanyol. Konfigurasi yang dipakai dalam proses pretraining sama dengan *vanilla* ALBERT. ALBETO memberikan 5 versi model, yaitu tiny, base, large, xlarge, dan xxlarge. Proses pretraining dilakukan dengan single TPU v3-8. Hasil evaluasi ALBETO pada tugas NLI (*Natural Language Inference*) dengan menggunakan *benchmark* XNLI, model ALBETO-xxlarge menghasilkan skor akurasi sebesar 82,42, sedangkan ALBETO-tiny menghasilkan skor akurasi yang paling kecil, yaitu 73,43.

Berdasarkan penelitian-penelitian yang telah diuraikan dalam Tabel 1.1, peneliti memilih untuk menggunakan model ALBERT (*A Little BERT*) karena masih relevan dalam riset inferensi bahasa dan deep learning. Selain itu ALBERT juga belum pernah digunakan untuk melakukan tugas NLI dengan data

ContractNLI. Maka dari itu, keaslian penelitian ini dapat dipertanggungjawabkan.

BAB 2

KAJIAN PUSTAKA

2.1 Tinjauan Pustaka

Chalkidis dkk., (2017) membuat benchmark dataset yang berisi 3.500 kontrak berbahasa Inggris, dan dikategorikan secara manual ke dalam 11 tipe elemen kontrak. Beberapa kata dalam dataset telah diubah kedalam kode angka karena masalah privasi. Contohnya kata “*termination*” diubah menjadi “3156”. Pada dataset, 993 kontrak dengan tipe clause headings dibagi menjadi 893 data training dan 100 data testing. Sedangkan 10 tipe sisanya terkandung dalam 2461 data yang dibagi menjadi 2111 data training dan 350 data testing. Dataset didesain untuk melatih model dalam mengekstrak elemen dalam kontrak secara otomatis. Elemen tersebut diantaranya judul, pihak yang terikat, tanggal efektif, tanggal selesai, hukum yang mendasari, dan periode kontrak. Untuk mengekstrak fitur dari dataset, digunakan 2 (dua) metode untuk membuat embedding. Pertama adalah manual (*hand crafted*), yang kedua adalah menggunakan Word2Vec (Mikolov dkk., 2013). Metode yang digunakan untuk mengenali elemen kontrak adalah *Sliding Window* dengan *Linear Logistic Regression* dan *Linear SVM (Support Vectore Machine)*. Hasil eksperimen menunjukkan bahwa sliding window dengan SVM dan kombinasi fitur manual dan Word2Vec memberikan performa paling maksimal, dengan macro-average precision 0,76, recall sebesar 0,86, dan F1 sebesar 0,80.

Berbeda dari penelitian sebelumnya, Xiao dkk., (2018) mengenalkan CAIL2018 (*Chinese AI and Law Challenge Dataset*), merupakan dataset skala besar pertama yang berfokus pada tugas memprediksi putusan hakim. Berisi lebih dari 2,6 juta kasus kriminal yang dipublikasi oleh Mahkamah Agung Rakyat Cina. Metode yang digunakan pada penelitian ini adalah TF-IDF + SVM, FastText, dan CNN (Convolutional Neural Network). Data dipecah menjadi 1.710.856 untuk training dan 965.219 untuk testing. Untuk proses training, optimizer yang digunakan adalah Adam (Kingma & Ba, 2014), learning rate yang digunakan

adalah 0,001, dropout rate 0,5, dan batch size 128. Hasil dievaluasi dengan metrik *Acc (Accuracy)*, *MP (Macro-Precision)*, dan *MR (Macro-Recall)*. Hasilnya CNN menunjukkan performa paling tinggi. Memiliki *Acc* 97,6 % dalam memprediksi dakwaan, *Acc* 97,6% dalam memprediksi undang-undang, dan *Acc* 78,2% dalam memprediksi ketentuan penalti.

Leivaditi dkk., (2020) merancang dataset yang berisi 179 dokumen kontrak sewa properti yang didesain untuk tugas mengidentifikasi entitas dan ketidaknormalan (*redflag*) secara otomatis. Entitas pada kontrak dapat berupa pihak yang berkepentingan, tanggal, jumlah uang, hak dan kewajiban secara spesifik, atau peraturan pemerintah yang terlibat. Sedangkan ketidaknormalan dapat berupa kalimat atau istilah yang berpotensi menimbulkan masalah pada pihak yang terikat kontrak. Model yang digunakan berbasis dari ALBERT (Lan dkk., 2020) dan dinamakan ALeaseBERT [elisbert]. Pada tugas identifikasi *redflag*, satu dense layer dengan keluaran softmax ditambahkan ke dalam ALBERT, lalu digunakan optimizer Adam, dan dilatih dalam 10 epoch. Pada tugas identifikasi entitas, satu layer klasifikasi token ditambahkan ke dalam ALBERT, dengan Adam sebagai optimizer, dan dilatih dalam 5 epoch. Hasilnya pada identifikasi *redflag*, ALeaseBERT memberikan nilai *MAP (Mean Average Precision)* sebesar 0,57 dan *IP@R=0,8 (Interpolation Precision on Recall 0,8)* sebesar 0,35. Pada tugas identifikasi entitas, hasilnya adalah *Precision MA (Micro Avg)* 0,5, *Recall MA* 0,48, *F1 MA* 0,4, *Precision WA (Weighted Avg)* 0,6, *Recall WA* 0,48, *F1 WA* 0,54.

Hendrycks dkk., (2021) mengenalkan CUAD (*Contract Understanding Atticus Dataset*), dataset yang berisi 500 kontrak dan telah ditinjau oleh ahli hukum untuk mengidentifikasi 41 tipe klausa (label), dan lebih dari 13.000 anotasi. Dataset ini didesain untuk tugas klasifikasi, dimana model dituntut untuk mengidentifikasi potongan kalimat dari dataset dan menentukan kategori dari kalimat tersebut. Model yang diuji terhadap dataset ini adalah BERT (Devlin dkk., 2019), RoBERTa (Liu dkk., 2019), ALBERT (Lan dkk., 2020), dan DeBERTa (He dkk., 2020). Dengan menggunakan metrik *Precision*, *Recall*, dan *AUC-PR (Area Under the Curve Precision Recall)*, model dengan performa terbaik adalah DeBERTa-xlarge, menghasilkan *AUC-PR* sebesar

47,8%, nilai precision 44,0% dengan recall sebesar 80%, dan nilai precision 17,8% dengan recall sebesar 90%. Pada kasus CUAD, nilai recall lebih penting dari nilai precision. Proses training dilakukan dengan learning rate dipilih dari $\{3 \times 10^{-5}, 1 \times 10^{-4}, 3 \times 10^{-4}\}$, besar epoch dipilih dari $\{1, 4\}$. Pemecahan data training dan testing dilakukan secara acak dengan pembagian 80% dan 20%. Optimizier yang digunakan adalah Adam (Kingma & Ba, 2014). Proses training dilakukan dengan menggunakan 8 A100 GPU.

2.2 Landasan Teori

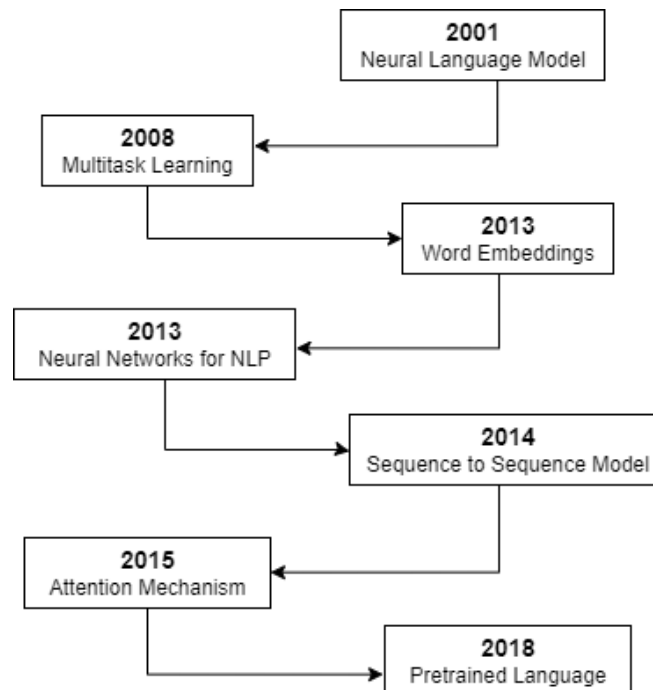
2.2.1 *Natural Language Processing (NLP)*

Natural Language Processing (NLP) adalah area yang berfokus pada teknik berbasis komputasi dan kecerdasan buatan untuk menganalisis dan merepresentasikan bahasa manusia dan hubungannya dengan komputer (Khan dkk., 2023). Model-model NLP dan algoritmanya dapat menemukan kesamaan, mengekstrak informasi penting dari informasi-informasi yang tidak penting, memperkirakan hasil dan tipologi berdasarkan representasi semantik dari teks, hingga membuat teks baru dari informasi yang diberikan dengan sedikit campur tangan manusia (Just, 2024). Perkembangan NLP terkini telah menunjukkan progres yang bagus dalam tugas-tugas bahasa seperti klasifikasi, tanya-jawab, mesin penerjemah, dan pengenalan suara (Khan dkk., 2023). Berdasarkan survey McKinsey, satu-per-tiga dari semua AI (*Artificial Intelligence*) yang dipakai mengimplementasikan NLP ke dalam produk atau proses bisnisnya, membuat NLP menjadi salah satu teknologi AI yang paling banyak digunakan, setara dengan robotika dan pengolahan citra (Chui dkk., 2022).

Pada akhir 1940an istilah NLP belum lahir. Pekerjaan yang melibatkan mesin dan bahasa manusia hanya MT (*Machine Translation*), dan pekerjaannya didominasi oleh bahasa Inggris dan Rusia. Bahkan berdasarkan report oleh ALPAC, pada 1966 NLP/MT hampir dinyatakan mati karena tidak ada progres yang signifikan (Khurana dkk., 2023). Namun dengan munculnya mesin yang lebih baik dengan kemampuan

komputasi lebih tinggi, berbagai penelitian NLP mulai populer kembali. Hingga saat ini NLP terus berkembang dengan munculnya inovas-inovasi baru dengan sangat cepat, seperti yang terlihat pada Gambar 2.1.

Saat ini NLP mengaplikasikan model statistik, machine learning, dan deep learning untuk mengenali pola pada bahasa alami dan mengubahnya sesuai keinginan dengan bantuan komputasi linguistik (Just, 2024). NLP dapat dibagi ke dalam 2 (dua) kategori, yaitu *natural language understanding* dengan tujuan menganalisa struktur grammar dan semantik dari teks untuk memahami maknanya dan *natural language generation* yang bertujuan untuk membuat teks secara otomatis berdasarkan input khusus yang diberikan (Kavlakoglu, 2020).

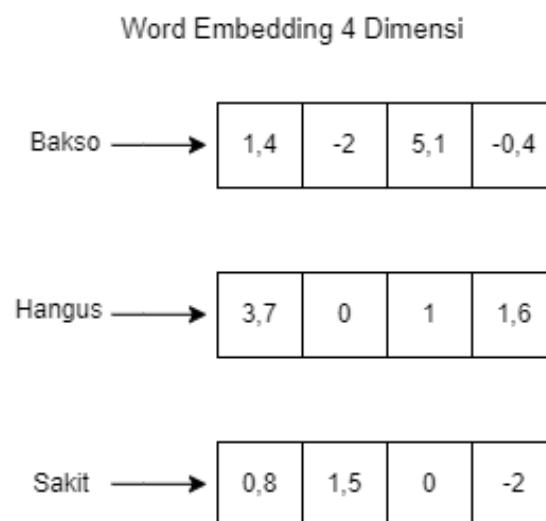


Gambar 2. 1 Perkembangan terkini dalam NLP

Sumber: (Khurana dkk., 2023)

Word Embedding adalah teknik yang digunakan untuk merepresentasikan kata-kata sebagai vektor angka dengan panjang tertentu dengan memetakan kata-kata ke dalam vektor, memungkinkan pengelompokan kata dengan konteks yang sama (Khan dkk., 2023).

Seperti yang diilustrasikan dalam Gambar 2.2, dalam NLP, teknik ini digunakan untuk mengekstrak fitur dari teks masukan dan menangkap informasi semantik dan sintatik dari teks tersebut. Beberapa tugas dalam NLP yang diuntungkan dengan adanya word embedding diantaranya, PoS (*Part of Speech*) Tagging, klasifikasi sentimen, translasi mesin, NER (*Named Entity Recognition*), analisa teks, analisa sintaks, colloquial analysis, tanya-jawab, keterikatan tekstual, dan sistem-sistem pada IoT (*Internet of Things*). *Word Embedding* terbagi ke dalam 2 (dua) kategori, yaitu *context-dependent* dan *context-independent* (Khan dkk., 2023). *Word Embedding* klasik bersifat *context-independent* yang menggunakan LM (*Language Model*) didasarkan pada arsitektur neural network sederhana, seperti GMM (*Gaussian Mixture Models*), SVM (*Support Vector Machine*), Logistic Regression, MLP (*Multi-Layer Perceptron*), dan CRF (*Conditional Random Fields*). *Word Embedding* dengan sifat *context-dependent* merupakan kebalikannya, karena dibuat dengan arsitektur model *Deep Learning* dan dapat membedakan embedding untuk kata yang sama berdasarkan perbedaan informasi atau konteks yang digunakan.



Gambar 2. 2 *Word Embedding* dengan vektor 4 dimensi

Sebelum digunakannya pendekatan DL (*Deep Learning*) dalam NLP, digunakan pendekatan tradisional dengan model berdimensi tinggi,

fitur sedikit, dan melakukan komputasi secara linear. Contoh dari model linear tersebut adalah SVM, Logistic Regression, dan CRF (Khan dkk., 2023). Di era modern ini, penelitian dalam NLP memanfaatkan DL dan memberikan performa yang terus bersaing tiap tahunnya. Banyak tugas-tugas NLP seperti klasifikasi, PoS, NER, translasi mesin, dan lain-lain yang berpindah dari model linear ke model DL, karena lebih superior dalam konteks performa. Kelebihan utama dari model DL adalah, dapat mempelajari data mentah dan tak berlabel secara otomatis, dan tak membutuhkan feature engineering yang spesifik hanya ke tugas tertentu saja (Goldberg, 2016). Berikut adalah beberapa tugas yang ada dalam NLP. Tugas PoS (*Part of Speech*) *Tagging* merupakan proses dimana pelabelan diaplikasikan ke dalam fitur yang terdiri dari kelompok linguistik berbeda. PoS berfungsi untuk menentukan label untuk tiap kata dalam teks yang diberikan (Khan dkk., 2023).

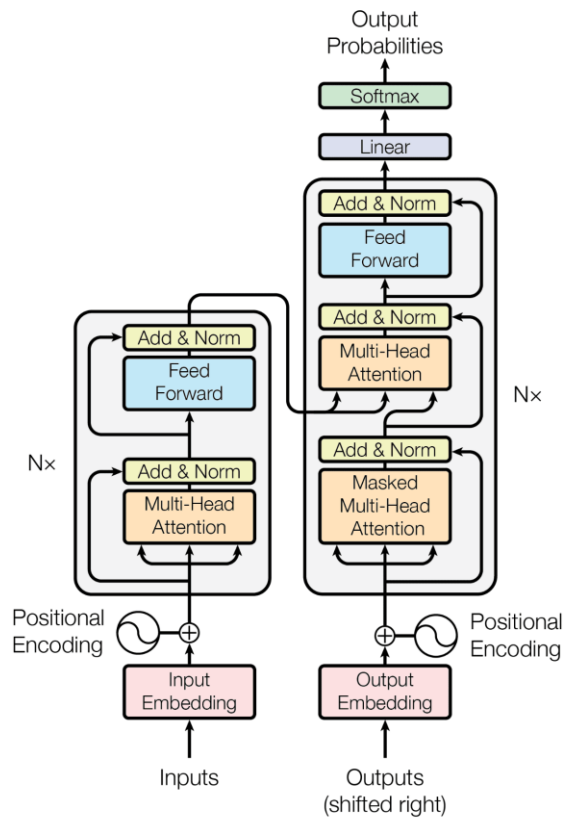
NER (*Named Entity Recognition*) adalah proses dimana mesin dapat mengidentifikasi dan membedakan kata dalam teks berdasarkan kelompok kata benda seperti orang, tempat, organisasi, waktu, jumlah, nilai uang, dan seterusnya (Khan dkk., 2023). NER terkadang disebut juga sebagai Sequential labeling task. NER merupakan salah satu proses krusial dalam tugas NLP lain seperti dialog, tanya-jawab, atau speech recognition.

Kategorisasi teks atau klasifikasi teks adalah proses untuk menentukan kategori dari teks dalam dokumen secara otomatis berdasarkan kategori-kategori yang telah ditentukan sebelumnya (Khan dkk., 2023). Beberapa contoh dari klasifikasi teks adalah penyaringan spam, kategorisasi subjek dalam teks, analisa sentimen, identifikasi bahasa, dan klasifikasi genre.

Translasi mesin adalah proses mengubah teks dari satu bahasa ke dalam bahasa lain dengan memanfaatkan teknik otomatisasi. Karena adanya ambiguitas and fleksibilitas dalam bahasa manusia, translasi mesin merupakan salah satu tugas yang menantang dalam NLP (Khan dkk., 2023). Saat ini, pendekatan dalam pembuatan translasi mesin adalah rule-based, machine learning-based, dan DL-based.

2.2.2 Transformer

Transformer adalah model *sequence-to-sequence* yang sepenuhnya dibangun dengan lapisan attention, menggantikan model dengan lapisan *recurrent* yang sebelumnya paling banyak digunakan (Vaswani dkk., 2017). Dalam tugas translasi mesin, Transformer memiliki performa yang secara signifikan lebih baik dari arsitektur model yang menggunakan lapisan recurrent atau convolutional. Transformer menggunakan arsitektur encoder-decoder seperti yang diilustrasikan dalam Gambar 2.3. Bagian encoder bertugas untuk memetakan rangkaian masukan dengan simbol (x_1, \dots, x_n) ke dalam rangkaian dengan simbol $z = (z_1, \dots, z_n)$. Lalu bagian decoder akan mengambil rangkaian z , dan mengubahnya ke dalam rangkaian keluaran (y_1, \dots, y_n) satu per satu (Vaswani dkk., 2017). Pada setiap step atau saat mengubah parameter, model Transformer bersifat *auto-regressive*, dimana simbol keluaran yang dikeluarkan sebelumnya dijadikan masukan tambahan saat menghasilkan rangkaian input akhir.



Gambar 2. 3 Arsitektur encoder-decoder Transformer

Sumber: (Vaswani dkk., 2017)

Lapisan attention pada Transformer menggunakan mekanisme berbeda yang dinamakan *Scaled dot-product attention*. Seperti yang terlihat pada Persamaan 2.1, Input dari mekanisme ini adalah queries Q dan keys K dengan dimensi d_k , dan values V dengan dimensi d_v . Dot product antara query dan key dihitung, lalu dibagi dengan $\sqrt{d_k}$, dan diaplikasikan fungsi softmax untuk mendapatkan nilai dari *value*.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (2.1)$$

$$Q = h_d W_q$$

$$K = h_e W_k$$

$$V = h_e W_v$$

$$h_d = \text{decoder states}$$

$$W_q = \text{query weight matrix}$$

$$W_k = \text{key weight matrix}$$

$h_e = \text{last encoder hidden states}$

Mekanisme scaled dot-product attention secara praktis lebih cepat dan lebih efisien ruang dibanding mekanisme attention lain seperti *additive attention* dan *dot-product attention*, karena mekanisme ini dapat diimplementasikan dengan kode perkalian matriks yang dioptimalkan (Vaswani dkk., 2017). Transformer menggunakan lebih dari satu lapisan attention yang ditumpuk dan dieksekusi secara paralel, dinamakan *Multi-Head Attention*. Formula dari *Multi-Head Attention* dapat dilihat pada Persamaan 2.2 dan 2.3.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_n)W^O \quad (2.2)$$

$$\text{dimana } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2.3)$$

$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$$

$$W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$$

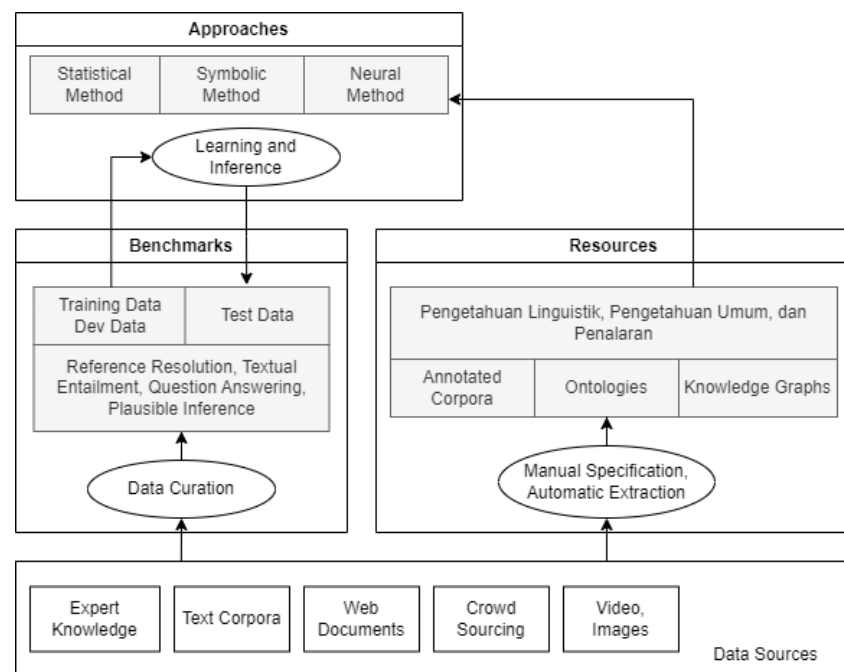
$$W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

$$W_i^O \in \mathbb{R}^{d_{\text{model}} \times h d_v}$$

Pada versi asli Transformer, jumlah head atau h yang digunakan berjumlah 8. Pada tiap h terdapat $d_k = d_v = \frac{d_{\text{model}}}{h} = 64$. Karena dimensi dalam tiap h dikurangi, maka secara komputasi akan sama ringannya dengan mekanisme yang menggunakan satu h atau *single-head attention* dengan dimensi penuh (Vaswani dkk., 2017). Transformer telah meningkatkan performa akurasi hampir semua LM (Language Models) dalam tugas NLP dan pemahaman bahasa (Wolf dkk., 2020). Pengaplikasiannya merentang dari pemodelan bahasa untuk membuat rangkaian teks secara otomatis, teks klasifikasi, analisa sentimen, tanya-jawab, memahami bacaan, NER, Pelabelan, translasi, dan penyimpulan teks. Banyak pula model yang dibangun dari arsitektur Transformers seperti GPT (Openai dkk., 2018) dan XLNet (Yang dkk., 2019) yang berfokus pada *autoregressive*, BART (Lewis dkk., 2019) dan T5 (Raffel dkk., 2019) yang berfokus pada *sequence-to-sequence*, RoBERTa yang berfokus pada multi-bahasa, dan DistilBERT (Sanh dkk., 2019) yang berfokus pada efisiensi.

2.2.3 Natural Language Inference (NLI)

Natural Language Inference (NLI) atau merujuk pada kemampuan mesin dalam memahami bahasa secara mendalam dan tak hanya apa yang diungkapkan secara eksplisit, melainkan mengandalkan kesimpulan yang diambil dari pengetahuan tentang bagaimana dunia bekerja (Storks dkk., 2020). Penelitian NLI dalam pengolahan bahasa alami dibagi ke dalam 3 area utama seperti pada Gambar 2.4, yaitu benchmarks yang menjadi tolak ukur seberapa bagus model, *knowledge resource* atau metode dalam menarik pengetahuan dari luar, dan terakhir adalah *approaches* atau pendekatan model dalam mempelajari dan memahami bahasa (Storks dkk., 2020).



Gambar 2. 4 Tiga area penelitian NLI dalam komunitas *Natural Language Processing*

Sumber: (Storks dkk., 2020)

Salah satu contoh kalimat yang didesain untuk tugas NLI berasal dari *Winograd Schema Challenge* (Levesque dkk., 2012): “The trophy would not fit in the brown suitcase because it was too big. What was too big?”. Tanpa adanya pengetahuan dari luar, mesin tak akan mengetahui

apa yang terlalu besar, piala atau koper. Pengetahuan dari luar yang dibutuhkan adalah “Untuk bisa masuk ke dalam objek A, objek B harus berukuran lebih kecil”. Tugas menalar dan pengambilan kesimpulan secara logis ini sangat mudah untuk manusia namun sulit dipelajari oleh komputer.

Storks dkk., merumuskan 3 (tiga) tipe pengetahuan yang diperlukan komputer untuk melakukan tugas NLI yang baik, yaitu:

- Pengetahuan Linguistik. Memahami makna gramatikal, struktur, semantik dan sintaks kata.
- Pengetahuan Umum. Fakta tentang dunia yang diketahui secara luas dan biasanya tertulis.
- Penalaran. Hal yang biasanya tidak tertulis dan diketahui secara umum oleh manusia.

Sejak munculnya *The Winograd Schema Challenge* oleh (Levesque dkk., 2012), telah banyak dataset yang dirancang khusus untuk NLI, seperti Event2Mind (Rashkin dkk., 2018), NarrativeQA (Kočiský dkk., 2017), MCScript (Osternann dkk., 2019), GLUE (Wang dkk., 2018), SWAG (Zellers dkk., 2018), dan CommonsenseQA (Talmor dkk., 2022). Dataset tersebut juga memiliki banyak tipe tantangan yang didesain untuk mengukur seberapa baik pemahaman komputer terhadap bahasa manusia, dari tanya-jawab, penarikan kesimpulan, sampai memahami emosi dan maksud melalui perilaku. Namun salah satu yang paling populer adalah tipe textual entailment.

Dataset atau yang biasa disebut *benchmark* dalam NLI dikelompokkan kedalam beberapa tipe. Tipe pertama adalah *Reference Resolution*, yang merupakan proses mengidentifikasi referensi atau sebutan di dalam teks yang ditujukan untuk pronoun atau frasa tertentu. Proses ini dapat menjadi sangat rumit karena adanya ambiguitas yang muncul karena adanya banyak entitas berpronomina dalam kalimat yang menuntut dibutuhkan pengetahuan lain dari luar (penalaran) untuk membuat keputusan (Storks dkk., 2020). Salah satu benchmark yang populer dalam tipe ini adalah *Winograd Schema Challenge* (WSC) dan

WinoGrade. Pada WSC, sistem disajikan dengan pertanyaan tentang kalimat berformat Skema Winograd. Untuk menjawab pertanyaan ini, sistem perlu membedakan pronoun yang merujuk pada salah satu entitas dan dapat berubah dengan mengganti satu kata dalam kalimat. Sedangkan WinoGrande merupakan versi peningkatan dari WSC dengan lebih fokus pada psikologi. Benchmark tersebut didesain sedemikian rupa supaya sistem tidak bisa menyelesaikannya hanya dengan memanfaatkan konteks linguistik. Meskipun begitu, mayoritas *Reference Resolution* benchmark hanya didesain untuk entitas, bukan *event* atau peristiwa. Harapannya, semakin banyak benchmark untuk *event reference resolution* yang bermunculan untuk meningkatkan progres NLI di dunia AI.

Tipe lain adalah *Question Answering* (QA). Berbeda dengan tipe sebelumnya, QA menuntut sistem untuk mempelajari beberapa kemampuan pemrosesan bahasa untuk menyelesaikan satu tugas. Mesin perlu memahami konteks dan entitas yang terdapat dalam pertanyaan, lalu membuat kesimpulan tentang keterikatannya dengan salah satu jawaban yang disediakan (Storks dkk., 2020). Walau begitu, tidak semua benchmark bertipe QA membutuhkan pengetahuan dari luar atau penalaran. Contoh beberapa benchmark dengan tipe QA adalah MCScript (2018), CoQA (2018), SQuAD (2016), dan CommonsenseQA (2019). MCScript dataset berisi 14.000 pertanyaan pilihan ganda yang sebagian besarnya hanya membutuhkan penalaran dasar untuk dijawab. Setiap pertanyaan disertai dengan paragraf pendek sebagai informasi tambahan. CoQA merupakan dataset yang berisi tanya-jawab dengan forma percakapan, dengan jumlah data sampai 127.000 pertanyaan. Sedangkan SQuAD menyajikan pertanyaan yang dilengkapi dengan paragraf sebagai sumber informasi. Dataset ini memiliki jumlah data sekita 150.000 pertanyaan dan jawaban.

Tipe selanjutnya adalah *Plausible Inference* yang menuntut sistem untuk merumuskan kesimpulan hipotetikal yang didasarkan dari konteks terbatas. Untuk melakukan tugas ini, sistem lebih membutuhkan penalaran dari pemahaman konteks linguistik atau pengetahuan eksternal.

Pengetahuan tersebut adalah penalaran dari interaksi sehari-hari seperti membuka pintu, memasak telur, atau berlari di musim panas. Beberapa benchmark dataset yang bertipe ini diantaranya adalah COPA (2011), JOCI (2017), ROCStories (2016), dan AlphaNLI (2019). *Benchmark COPA* mengevaluasi kemampuan penalaran kasual yang membutuhkan pengetahuan umum tentang hal yang terjadi di dunia nyata. Setiap contoh yang disajikan COPA, menyediakan premis dan menuntut sistem untuk menentukan penyebab atau efeknya dari dua pilihan, dengan begitu dapat menguji kemampuan penalaran ke masa depan dan masa lalu. *Benchmark JOCI* terdiri dari 39.000 pasang kalimat, yang terdiri dari konteks dan hipotesis. Sistem dituntut untuk menentukan rating 1 sampai 5 atas seberapa cocoknya pasangan konteks-hipotesis ini. Kasus ini hampir mirip seperti NLI keterikatan tekstual dengan 3 kelas, namun bedanya JOCI menyediakan lebih banyak opsi antara *contradict* dan *entail*. *Benchmark ROCStories* berisi 50.000 data yang terdiri dari 5 kalimat dari kehidupan sehari-hari. Kalimat tersebut mengandung hubungan sebab-akibat dari suatu peristiwa. Dari jumlah tersebut, 3.700 diantaranya didesain sebagai kasus uji, dengan format yang disebut *Story Cloze Test*. Sedangkan benchmark AlphaNLI menguji sistem dengan cara menyediakan 2 (dua) observasi dan sistem dituntut untuk menentukan hipotesis yang masuk akal berdasarkan hasil observasi tersebut. Berbeda dengan benchmark-benchmark sebelumnya, AlphaNLI menuntut untuk dapat melakukan penalaran dari kalimat (observasi) tanpa implikasi penalaran yang sebelumnya telah diberikan. AlphaNLI terdiri dari 20.000 kalimat konteks dan 200.000 hipotesis, dan saat dipasangkan dapat membentuk 170.000 pasangan untuk subset *training* dan *development* (Storks dkk., 2020).

2.2.4 Textual Entailment

Textual entailment (keterikatan tekstual) adalah hubungan searah antara teks (premis) dan hipotesis, dimana dapat disimpulkan bahwa status entails dicapai jika orang dapat menyimpulkan bahwa hipotesis benar

berdasarkan teks (premis) yang diberikan (Dagan dkk., 2006). Keterikatan tekstual merupakan contoh tipe lain dari NLI, bersama dengan QA, *reference resolution*, dan *plausible inference*. Berikut adalah contoh pasangan premis-hipotesis textual entailment dari dataset SNLI (Bowman dkk., 2015) :

Tabel 2. 1 Sampel kasus dalam dataset S NLI (*Stanford Natural Language Inference*)

Premis	Hipotesis	Status
<i>A soccer game with multiple males playing.</i>	<i>Some men are playing a sport.</i>	<i>entail</i>
<i>A man inspects the uniform of a figure in some East Asian country.</i>	<i>The man is sleeping</i>	<i>contradiction</i>
<i>An older and younger man smiling.</i>	<i>Two men are smiling and laughing at the cats playing on the floor.</i>	<i>neutral</i>

Layaknya tugas-tugas lain dalam NLI, untuk menyelesaikan tugas keterikatan tekstual dibutuhkan kemampuan-kemampuan sederhana dari sebuah pemrosesan bahasa seperti NER dan coherence resolution. Namun, berbeda dengan tugas tanya-jawab, pada keterikatan tekstual kemampuan menalar adalah yang paling penting (Storks dkk., 2020). Beberapa benchmark untuk mengukur kemampuan model dalam tugas keterikatan tekstual diantaranya adalah RTE (*Recognizing Textual Entailment*) *Challenges* yang dibuat dengan tujuan untuk mengevaluasi kemampuan mesin dalam memahami pengetahuan umum dan penalaran, *Conversational Entailment* dengan format dialog sebagai premis, SICK (*Sentences Involving Compositional Knowledge*) (Zhang dkk., 2018), SNLI (*Stanford Natural Language Inference*) (Glockner dkk., 2018), SciTail (Khot dkk., 2018) yang berfokus pada domain sains, dan SherLLiC (Schmitt & Schütze, 2019) yang berfokus pada hubungan antar kata.

2.2.5 Surat Kontrak

Surat kontrak adalah aspek penting dalam dunia komunikasi bisnis, dan melibatkan surat yang memiliki nilai legalitas dan dapat membentuk basis dari persetujuan atau perselisihan (GW University, 2023). Tujuan dari surat kontrak adalah untuk menetapkan kesepakatan secara tertulis antara beberapa pihak. Satu pihak dapat mengirim surat kepada pihak lainnya untuk mengajukan, negosiasi, atau menerima ketentuan di dalam surat kontrak. Satu pihak juga dapat mengirim surat untuk menegaskan hak-hak pihak lain, menuntut pembayaran, atau memberikan peringatan atas pelanggaran yang telah dilakukan. Surat kontrak harus bersifat objektif, formal, dan profesional. Surat tersebut harus mencerminkan pendekatan bisnis yang benar dan menghindari bahasa yang terlalu emosional. Struktur dari surat kontrak adalah sebagai berikut.

1. *Salutation*

Surat harus diawali dengan nama penerima dan gelarnya.

2. *Opening Paragraph*

Paragraf ini harus mendeskripsikan tujuan dari surat kontrak.

3. *Body Paragraph*

Tubuh atau isi utama dari surat kontrak harus menguraikan perubahan, detail, kondisi, atau kebutuhan yang berhubungan dengan kesepakatan yang telah dibuat.

4. *Closing Paragraph*

Paragraf ini harus berisi ringkasan dari aksi yang harus pihak penerima lakukan, berkaitan dengan isi dari *body paragraph*.

5. *Closing Salutation*

Surat harus diakhiri dengan salam penutup yang sopan, diikuti dengan nama dan gelar pengirim.

Surat kontrak dapat dikategorikan ke dalam beberapa tipe sesuai dengan fungsi dan basis legalitasnya. Beberapa tipe tersebut adalah sebagai berikut.

1. *Affiliation Agreement*

Affiliation Agreement adalah kontrak antara satu pihak dengan pihak lain yang memiliki tujuan untuk membangun realasi dan berbagi sumber daya untuk tujuan tertentu.

2. *Amendment* (Modifikasi)

Amendment adalah modifikasi yang ditulis untuk kesepakatan kontrak yang telah ada dan telah disetujui. Secara formal, surat ini ditulis untuk menambah informasi atau mengubah kondisi di dalam kontrak. Saat *amendment* ditandatangani oleh semua pihak terikat, isi dari surat ini akan menjadi bagian dari surat kontrak sebelumnya yang dimodifikasi.

3. *Independent Contractor Agreement*

Independent contractor agreement atau biasa disebut *professional service agreement* digunakan ketika satu pihak ingin mengikat kontrak dengan penyedia layanan dari pihak lain yang bukan merupakan pegawai. Surat ini akan menentukan pihak mana yang dikategorikan sebagai kontraktor independen dan pegawai.

4. Sewa

Surat kontrak sewa adalah kontrak dimana pemilik *real estate*, fasilitas, atau perlengkapan memberikan hak penggunaan fasilitas tersebut kepada pihak lain, untuk jangka waktu yang spesifik dengan jumlah harga sewa yang spesifik pula.

5. *License*

License adalah kontrak yang memungkinkan pemilik memberikan izin kepada pihak lain untuk menggunakan sesuatu atau untuk mengizinkan aktivitas yang harusnya dilarang tanpa adanya izin tertulis tersebut.

6. *Letter of Agreement* (LoA)

Kontrak ini adalah tipe kontrak yang memiliki format dari surat biasa. Walaupun umumnya surat ini lebih pendek dari tipe kontrak lain, namun LoA tetap memiliki isi dari surat kontrak pada umumnya.

7. *Letter of Intent*

Jika LoA adalah surat kontrak yang secara legal mengikat, maka *Letter of Intent* adalah sebaliknya. Surat ini digunakan untuk menuliskan ringkasan dari rencana umum dari transaksi yang diajukan sebelum persetujuan yang mengikat dilakukan. Surat ini dipandang sebagai ekspresi ketertarikan dari satu pihak ke pihak lain.

8. *Memorandum of Understanding (MoU)*

Serupa dengan *Letter of Intent*, MoU dapat berperan sebagai pernyataan ketertarikan yang digunakan untuk menjadi dasar dari kesepakatan yang dibuat untuk mencapai tujuan bersama, namun tidak melibatkan uang atau adanya kondisi tertentu yang harus dipatuhi pihak tertentu. Jika MoU mengandung kondisi seperti pertukaran uang atau kewajiban yang harus dipatuhi, maka surat ini akan dianggap sebagai kontrak yang secara legal mengikat.

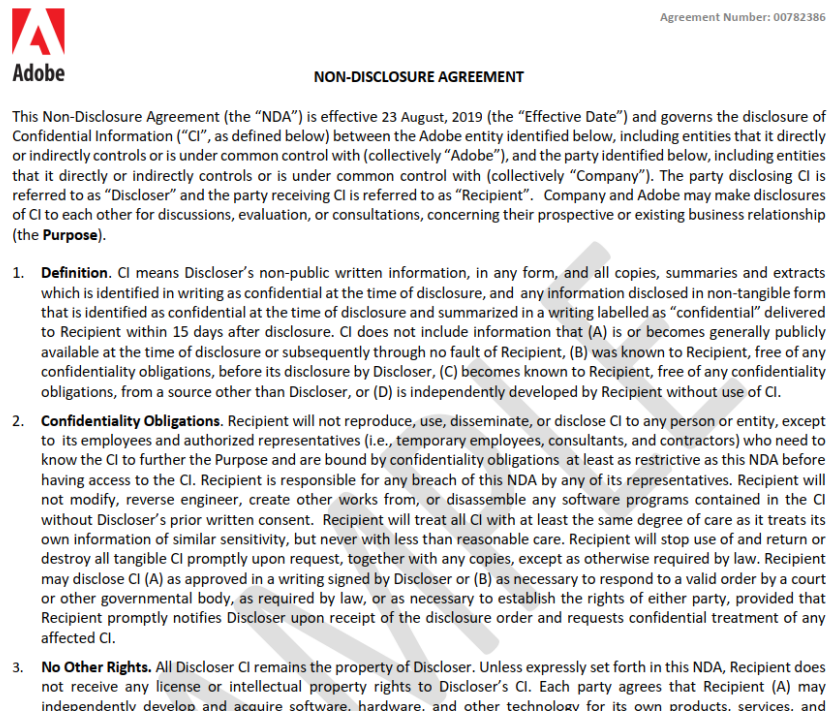
9. *Purchase Order (PO)*

Surat PO adalah surat yang ditulis oleh satu pihak kepada pihak lain untuk menyediakan sumber daya atau layanan. Surat ini mengandung syarat dan ketentuan yang akan mengatur transaksi pembelian. Saat PO diterima oleh kontraktor penyedia sumber daya atau jasa, surat kontrak lain akan dibuat yang menyediakan detail tentang pembayaran.

2.2.6 *NDA (Non-Disclosure Agreement)*

Surat perjanjian kerahasiaan atau *non-disclosure agreement* (NDA) adalah kontrak yang mengikat dua atau lebih pihak dimana beberapa atau semua pihak setuju bahwa informasi tertentu yang dibagikan dari pihak satu ke pihak lain atau dibuat oleh salah satu pihak akan bersifat rahasia (Radack, 1994). NDA memiliki beberapa fungsi. Pertama adalah melindungi informasi teknis atau komersial yang sensitif dari pihak lain. Kedua adalah mencegah pihak tertentu kehilangan hak paten dari sebuah penemuan atau inovasi. Ketiga adalah NDA mendefinisikan secara spesifik informasi apa yang dapat dan tidak dapat dibagikan (Radack, 1994). Jenis informasi yang dapat disertakan dalam NDA tidak terbatas,

dapat berupa prototipe, gambar arsitektur, perangkat lunak komputer, peralatan, hasil tes, ataupun daftar spesifikasi. Seperti yang terlihat pada Gambar 2.5, umumnya surat kontrak ini harus disertai periode waktu tertentu untuk menandakan kapan dimulai dan berakhirnya perjanjian kerahasiaan.



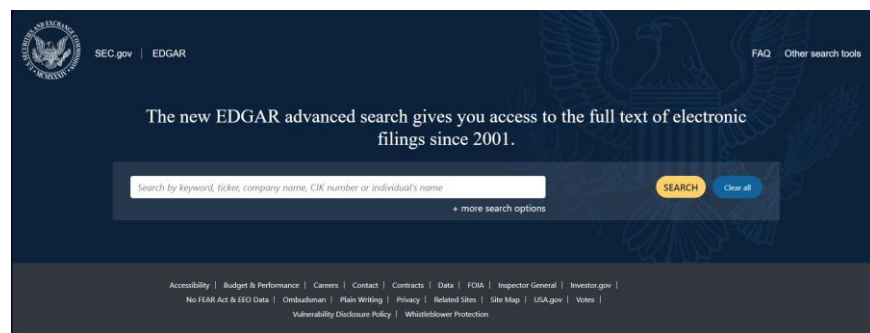
Gambar 2. 5 Contoh Surat Perjanjian Kerahasiaan (NDA)

Sumber: (Koreeda & Manning, 2021)

2.2.7 EDGAR

Electronic Data Gathering, Analysis, and Retrieval (EDGAR) adalah sistem pelaporan elektronik yang dibuat oleh badan independen Amerika *Security and Exchange Commission* (SEC) dengan tujuan untuk meningkatkan efisiensi dan aksesibilitas dalam mengakses file atau dokumen dari atau kepada SEC (Chen, 2022). Berdasarkan undang-undang sekuritas tahun 1993 di Amerika, semua perusahaan publik diwajibkan untuk mempublis data finansial setiap 3 (tiga) bulan. SEC mulai mengembangkan sistem pelaporan elektronik pada 1980 dan

meluncurkan versi awal pada 1984. Sistem EDGAR yang beroperasi sepenuhnya diluncurkan pada 1992. Dokumen-dokumen yang dilaporkan pada SEC melalui EDGAR meliputi laporan tahunan dan triwulan, informasi tentang kepemilikan investor institusi, laporan sejarah perusahaan, laporan keuangan yang diaudit, deksripsi produk dan layanan, tinjauan organisasi tahunan, dan pasar perusahaan. Database EDGAR mengandung lebih dari 20 tahun laporan elektronik. Pengguna dapat mencari dalam database layaknya mesin pencarian lain seperti pada Gambar 2.6. Hasil pencarian selanjutnya dapat dipersempit menggunakan filter berdasarkan tanggal, lokasi perusahaan, dan tipe file (Chen, 2022).



Gambar 2. 6 Sistem pencarian pada sistem EDGAR

Sumber: <https://www.sec.gov>

2.2.8 BERT

BERT (*Bidirectional Encoder Representations from Transformer*) adalah *pretrained bidirectional* model yang didesain untuk memahami konteks dalam teks dari dua arah pada tiap layer (Devlin dkk., 2019). Latar belakang dari dibuatnya BERT adalah kebanyakan *pretrained* model yang ada dibuat dengan teknik yang membatasi kemampuan dalam mempelajari konteks, terutama pada proses *finetuning*. Salah satu batasan ini adalah model bersifat *unidirectional*, dimana model hanya memahami konteks dari kiri ke kanan atau kanan ke kiri. Contoh dari model yang bersifat *unidirectional* adalah OpenAI GPT yang menggunakan arsitektur kiri ke kanan, dimana setiap token hanya dapat mempelajari token-token sebelumnya pada layer *self-attention* (Vaswani dkk., 2017). BERT

memanfaatkan dua metode dalam proses *pretraining* untuk melakukan *bidirectional training*, yaitu MLM (*Masked Language Model*) dan NSP (*Next Sentence Prediction*). MLM memiliki tujuan untuk melatih model untuk memahami teks dengan cara melengkapi kata yang hilang dari suatu sekuen teks. Sedangkan NSP menuntut model untuk belajar dengan cara memprediksi kalimat lanjutan dari suatu teks. BERT menyediakan dua tipe model, yaitu BERT-base dengan 110 juta parameter dan BERT-large dengan 340 juta parameter. Saat ini, sudah banyak versi model yang lahir dari model BERT. Beberapa diantaranya adalah DistilBERT (Sanh dkk., 2019), RoBERTa (Liu dkk., 2019), dan ALBERT (Lan dkk., 2020).

2.2.9 ALBERT

A Little BERT (ALBERT) merupakan arsitektur model yang didasarkan pada arsitektur BERT (Devlin dkk., 2019) dengan fokus pada efisiensi memori. ALBERT memiliki jumlah parameter yang secara signifikan lebih sedikit dari arsitektur BERT (Lan dkk., 2020), seperti yang terlihat pada Tabel 2.2. ALBERT-large memiliki parameter 18 kali lebih sedikit dari BERT-large. ALBERT-xlarge dengan hidden layer $H = 2048$ hanya memiliki 60 Juta parameter, sedangkan versi xxlarge dengan *hidden layer* $H = 4096$ hanya memiliki 233 Juta parameter, sekitar 70% dari jumlah parameter BERT-large. Dari desain model ini, dapat disimpulkan bahwa secara komputasi dan memori, ALBERT lebih efisien dari BERT.

Tabel 2. 2 Perbandingan model BERT dan ALBERT

Model		Params	Layers	Hidden State	Embedding	Param-sharing
BERT	base	108 Jt	12	768	768	<i>False</i>
	large	334 Jt	24	1024	1024	<i>False</i>
ALBERT	base	12 Jt	12	768	128	<i>True</i>
	large	18 Jt	24	1024	128	<i>True</i>
	xlarge	60 Jt	24	2048	128	<i>True</i>
	xxlarge	235 Jt	12	4096	128	<i>True</i>

ALBERT menggunakan 2 (dua) teknik untuk menurunkan jumlah parameter. Pertama adalah *factorized embedding parameterization*. Pada model BERT (Devlin dkk., 2019), XLNet (Yang dkk., 2019), dan RoBERTa (Liu dkk., 2019), ukuran *embedding* WordPiece E terikat dengan ukuran hidden layer H , dimana $E \equiv H$. Dari perspektif pemodelan, *embedding* WordPiece bertujuan untuk merepresentasikan data secara *context-independent*, sedangkan hidden layer bertujuan untuk mempelajari data secara *context-dependent*. Melepas ikatan antara ukuran *embedding* E dan hidden layer H memungkan efisiensi pada model, dimana $E \gg H$ (Lan dkk., 2020). Sedangkan dari perspektif praktikal, umumnya NLP menuntut kumpulan kosakata berukuran V sebesar mungkin. Karena ukuran hidden size H adalah $V \times E$, maka parameter model akan membengkak sementara parameter dalam hidden size yang diperbarui selama training hanya sebagian. Karenanya, ALBERT menurunkan ukuran parameter model dengan memfaktorisasi parameter *embedding*, dari $O(V \times H)$ menjadi $O(V \times E + E \times H)$. Teknik kedua untuk menurunkan ukuran parameter adalah *Cross-layer parameter sharing*. Ada 3 (tiga) cara untuk berbagi parameter, pertama adalah hanya berbagi parameter *feed-forward network* (FFN), hanya berbagi parameter attention, dan berbagi semua parameter. ALBERT memilih untuk berbagi semua parameter karena dirasa paling optimal (Lan dkk., 2020).

2.2.10 PyTorch

PyTorch adalah *Deep Learning library* yang dibuat dengan Python dan umumnya digunakan untuk pengembangan yang membutuhkan GPU dan CPU. *PyTorch* memanfaatkan komputasi graf dinamik yang memungkinkan saintis atau pengembang untuk menjalankan kode dan berganti antara CPU dan GPU secara *realtime* (Simplilearn, 2023). Fitur ini juga membuat proses *debugging* atau evaluasi kode baris demi baris menjadi lebih mudah. *PyTorch* memiliki beberapa modul yang paling sering digunakan, seperti *autograd*, *optim*, dan *nn*. Modul *autograd* adalah mesin diferensiasi otomatis yang membantu dalam perhitungan gradien di

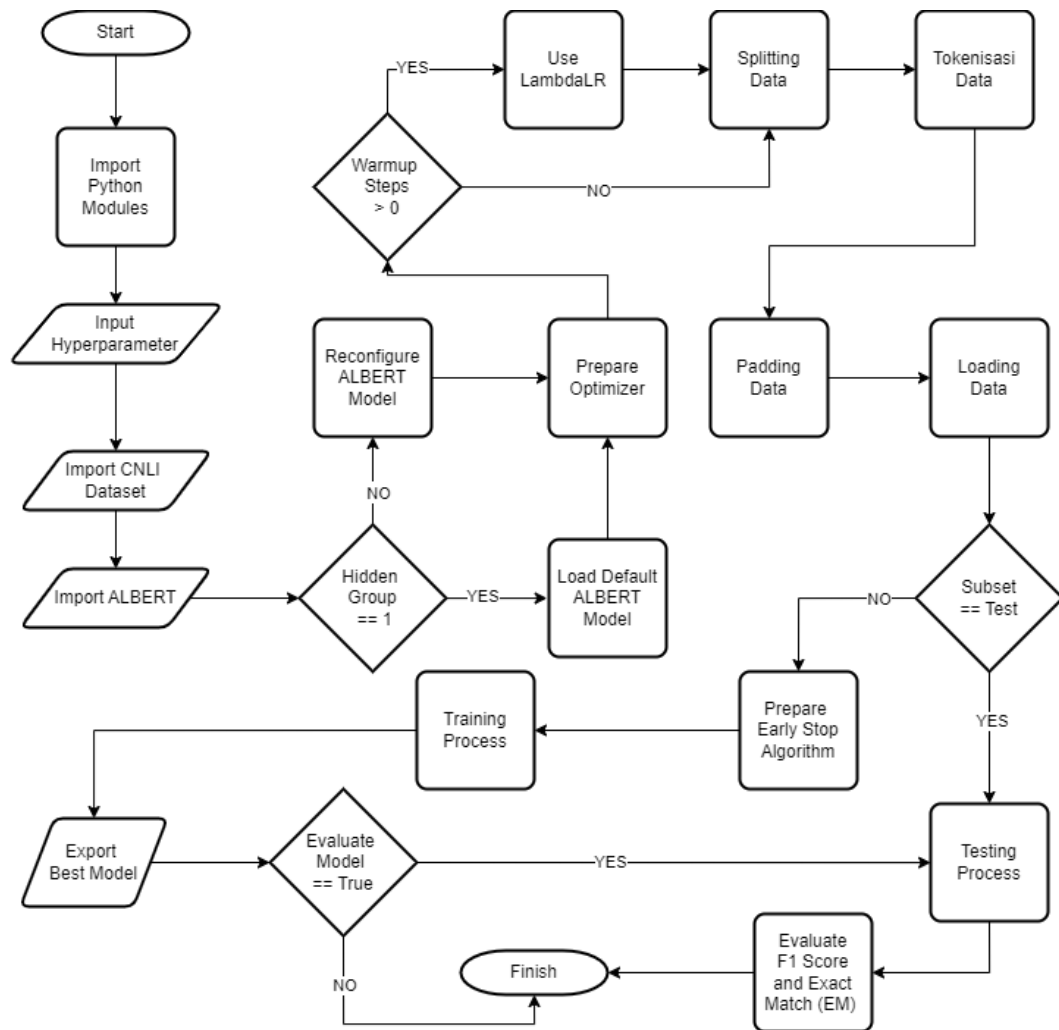
fase *forward pass*. Autograd membuat grafik asiklik terarah dimana daun adalah input tensor dan akarnya adalah output tensor. Modul optim adalah *package* yang menyediakan berbagai *optimizer* dan dapat langsung digunakan untuk kepentingan mengembangkan *neural network*. Modul nn terdiri dari berbagai *class* yang membantu dalam membangun neural network. Selain itu, *PyTorch* juga memiliki fitur bernama *DataLoader* dan *Dataset*, yang membantu pengembang dan saintis dalam menangani data dalam jumlah besar. *Dataset* dan *DataLoader* memungkinkan pengembang untuk menggunakan data mereka sendiri dan memecahnya sesuai ukuran batch, yang nantinya dapat diproses secara parallel oleh mesin.

BAB 3

METODE PENELITIAN

3.1 Pendekatan dan Desain Penelitian

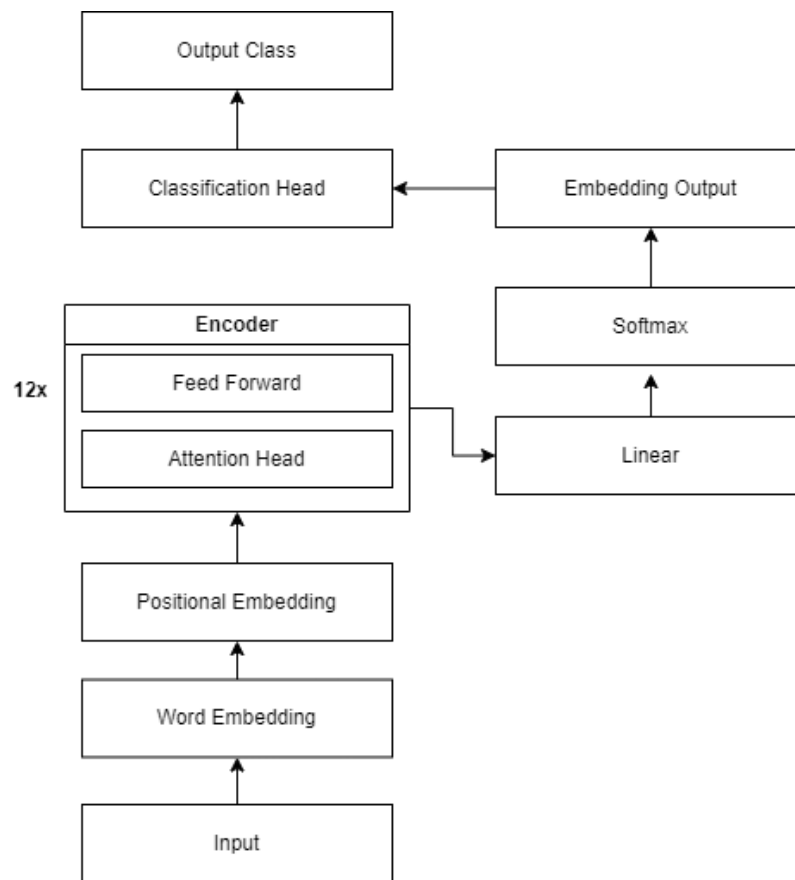
Penelitian tentang inferensi bahasa ini menggunakan metode *fine-tuning* dan *pre-training* dalam melatih model untuk memahami data CNLI. Alur kerja dari penelitian adalah seperti yang ditunjukkan pada Gambar 3.1. Awalnya modul yang dibutuhkan untuk proses *training* diimport. Setelah itu *hyperparameter* dikonfigurasi secara manual. Lalu masuk ke tahap *preprocessing* yang terdiri dari *splitting*, *tokenizing*, *padding*, dan *loading*. Setelah data dipersiapkan, arsitektur ALBERT dan cekpoinnya diambil dari halaman web *huggingface.co* untuk selanjutnya masuk ke tahap *fine-tuning*, dimana arsitektur layer model dikonfigurasi ulang untuk lebih menyesuaikan tugas yang akan dikerjakan. Selanjutnya *optimizer* dan *warmup steps* dipersiapkan untuk proses pelatihan (*training*). Terakhir proses pengujian (*testing*) dan evaluasi performa atau hasil dari model yang telah dilatih dengan data CNLI.



Gambar 3. 1 Alur kerja proses eksperimen peneliti

Model yang akan digunakan dalam penelitian ini adalah ALBERT (*A Little BERT*) yang merupakan peningkatan dari model BERT (Bidirectional Encoder Representations from Transformer). Peningkatan yang diberikan antara lain adalah usaha dalam mengurangi banyaknya parameter tanpa mengurangi performa secara signifikan. Untuk mencapai itu, ALBERT menggunakan 2 (dua) metode tambahan yaitu Factorized Embeddings Parameterization dan *Cross-layer Parameters Sharing*. Selain itu ALBERT juga menggunakan loss function berbeda yang bernama SOP (*Sentence Order Prediction*). Loss Function ini sangat menguntungkan dalam memproses teks panjang karena dapat mempertimbangkan

aliran logika dan hubungan antar-kalimat. Disamping itu, ALBERT dan BERT menggunakan arsitektur yang sama, seperti yang dapat dilihat pada Gambar 3.2.



Gambar 3. 2 Arsitektur model BERT dan ALBERT

Training akan dilakukan beberapa kali dengan memperhitungkan 2 (dua) tipe *training*, yaitu pre-training atau melatih semua layer ALBERT dengan data CNLI atau tanpa pre-training. Tiap tipe tersebut akan dilatih juga beberapa kali dengan hyperparameter berbeda. Hyperparameter yang diperhitungkan antara lain *batch size*, *hidden groups*, dan *early stops*. *Batch Size* melambangkan berapa banyak data yang akan dimasukkan ke dalam neural network dalam satu step. Satu step adalah proses dimana gradien pada neural network diperbarui untuk mendekati titik *convergence*. *Hidden groups* melambangkan banyak grup layer yang akan saling berbagi parameter. Ini merupakan ciri khas dari model ALBERT. Sedangkan *early steps* digunakan supaya proses training berhenti jika model sudah tidak menunjukkan progres menuju titik *convergence*. Situasi ini ditunjukkan dengan meningkatnya akurasi atau berkurangnya nilai loss.

3.2 Lokasi Penelitian

Penelitian dilakukan dalam ruang lingkup kampus Universitas Negeri Semarang yang berlokasi di Sekaran, Gunung Pati, Kota Semarang, Jawa Tengah. Tempat yang digunakan antara lain adalah laboratorium komputer gedung Digital Center, Perpustakaan Fakultas Matematika dan Ilmu Pengetahuan Alam, serta tempat tinggal peneliti yang berada tidak jauh dari Kampus.

3.3 Fokus Penelitian

Fokus dari penelitian ini adalah untuk melakukan tugas inferensi bahasa menggunakan data CNLI yang terfokus pada domain legal atau hukum dengan mengutamakan efisiensi memori, meminimalkan perangkat keras yang digunakan, dan membuat ukuran parameter model sekecil mungkin, namun masih tetap memaksimalkan akurasi untuk menyamai atau bahkan melebihi model basis (*base model*) Span NLI BERTbase oleh (Koreeda & Manning, 2021). Selain itu peneliti juga berfokus pada performa dari model yang diteliti dalam menentukan keterikatan tekstual dari premis dan hipotesis dalam konteks surat perjanjian kerahasiaan.

3.4 Data dan Sumber Data

Data CNLI (*Contract Natural Language Inference*) yang telah dibuat oleh (Koreeda & Manning, 2021) dan dirancang untuk melatih model dalam tugas inferensi bahasa dalam domain hukum dengan data berupa dokumen kontrak persetujuan kerahasiaan memiliki rata-rata token per dokumen sebesar 2.254, dimana model ALBERT yang digunakan dalam penelitian memiliki maksimum panjang token sebesar 512. Hal ini dikarenakan oleh data diambil langsung dari dokumen kontrak persetujuan kerahasiaan asli yang secara umum panjangnya lebih dari satu halaman. Dalam dataset CNLI, terdapat 607 dokumen yang dipilih secara manual dan 17 hipotesis yang dibuat secara manual oleh (Koreeda & Manning, 2021), seperti yang terlihat pada Gambar 3.3.



```

{
  "documents": [
    {
      "document_1": []
    },
    {
      "document_2": []
    },
    ...
    {
      "document_607": []
    }
  ],
  "labels": [
    {
      "nda_1": {
        "hypothesis": ""
      }
    },
    ...
    {
      "nda_17": {
        "hypothesis": ""
      }
    }
  ]
}

```

Gambar 3. 3 Struktur file JSON dari CNLI sebelum dilakukan *pre-processing*

Karena data CNLI yang disediakan berformat JSON dan JSONL, peneliti perlu melakukan penstrukturan ulang dan memecah struktur JSON pada Gambar 3.3 menjadi bentuk tabel dataset. Tabel dataset ini yang nantinya akan dipecah menjadi grup data *training*, *development*, dan *testing*, dengan persentasi 70%, 10%, dan 20%. Pembagian jumlah dokumen untuk proses training dapat dilihat pada Tabel 3.1. Pada tabel tersebut, dapat ditunjukkan pemecahan data yang dikelompokkan berdasarkan format pengumpulan oleh (Koreeda & Manning, 2021) dan sumber pengumpulannya. Pembahasan lebih detail mengenai sumber dan format pengumpulan data akan dibahas lebih rinci pada poin 3.5 Teknik Pengumpulan Data.

Tabel 3. 1 Pemecahan data berdasarkan sumber dan format dokumen kontrak

Format	Sumber	Train	Dev	Test	Total
Teks Biasa	EDGAR	83	12	24	119
HTML	EDGAR	79	11	23	113
PDF	Mesin Pencarian	261	38	76	375
Total		423	61	123	607

Sumber : (Koreeda & Manning, 2021)

Untuk merumuskan 17 hipotesis pada dataset CNLI, berbagai surat kontrak persetujuan kerahasiaan perlu dibandingkan dan hasilnya telah ditinjau ulang oleh spesialis hukum (Koreeda & Manning, 2021). Setiap hipotesis dikemas seperti yang ditunjukkan pada Gambar 9, dan didampingi dengan judul. Judul hipotesis hanya bertujuan supaya mudah dibaca oleh peneliti lain. Persebaran hipotesis relatif terhadap label entail dan contradiction dapat dikatakan tidak seimbang. Beberapa pasangan hipotesis dan premis memiliki label berstatus entail lebih banyak, beberapa memiliki label neutral lebih banyak, dan sebagian banyak pasangan memiliki sedikit bahkan sampai tidak memiliki label berstatus contradiction, seperti dapat dilihat pada Tabel 3.2.

Tabel 3. 2 Persebaran kelas berdasarkan hipotesis. (Judul hipotesis hanya bertujuan supaya mudah dibaca).

Judul Hipotesis	Entailment	Contradiction	Neutral
<i>1. Explicit identification</i>	220	112	91
<i>2. None-inclusion of non-technical information</i>	91	309	23
<i>3. Inclusion of verbally conveyed information</i>	149	4	270
<i>4. Limited use</i>	52	7	364
<i>5. Sharing with employees</i>	65	13	345
<i>6. Sharing with third-parties</i>	53	111	259
<i>7. Notice on compelled disclosure</i>	147	0	276
<i>8. Confidentiality of Agreement</i>	262	2	159

9. No reverse engineering	363	1	59
10. Permissible development of similar information	160	0	263
11. Permissible acquirement of similar information	112	0	311
12. No licensing	96	0	327
13. Return of confidential information	241	1	181
14. Permissible copy	256	76	91
15. No solicitation	330	0	93
16. Survival of obligations	112	11	300
17. Permissible post-agreement possession	111	194	118

Sumber: (Koreeda & Manning, 2021)

Sebelum dilakukan penelitian, data CNLI perlu di ubah strukturnya ke dalam tabel *dataframe* dengan memasang 607 dokumen dengan 17 hipotesis yang tersedia. Hasilnya, struktur tabel *dataframe* akan menjadi seperti yang diilustrasikan pada Tabel 3.3. Dengan begitu, total sampel data CNLI untuk proses penelitian akan menjadi $607 \times 17 = 10.319$ baris data.

Tabel 3. 3 Struktur tabel dataset dari ContractNLI

Kolom	Tipe Data	Keterangan
Premis	String	Teks surat kontrak
Hipotesis	String	Teks hipotesis
Label	String	["entail", "contradict", "not mentioned"]
Subset	String	["train", "test", "dev"]

3.5 Teknik Pengumpulan Data

Berdasarkan penelitian yang telah dilakukan oleh (Koreeda & Manning, 2021), dokumen kontrak (NDA) dikumpulkan dari sistem pencarian internet dan sistem EDGAR (*Electronic Data Gathering, Analysis, and Retrieval*). Pencarian dokumen dilakukan dengan menggunakan kata kunci dan regular expression dan

hasilnya dipilih secara manual untuk mendapatkan dokumen kontrak yang valid. Tipe surat kontrak (NDA) yang dipilih adalah kontrak bilateral dan unilateral, atau surat persetujuan kerahasiaan antar dua pihak. Tipe kontrak employee-employer dikecualikan karena isinya berbeda dengan surat kontrak lainnya dalam data.

Pengumpulan data menggunakan mesin pencarian Google dilakukan dengan kata kunci “*non-disclosure agreement filetype:pdf*”. Pencarian dilakukan dalam 7 (tujuh) domain negara berbeda yang menggunakan bahasa Inggris sebagai bahasa utama, yaitu Amerika Serikat, Britania Raya, Australia, Selandia Baru, Singapura, Kanada, dan Afrika Selatan. Sedangkan dokumen yang dikumpulkan menggunakan sistem EDGAR merupakan kontrak yang berasal dari arsip tahun 1996 sampai 2020. Selanjutnya konten dalam dokumen diekstrak dengan regular expression dengan pola “*(?<![a-zA-Z,\"()]*)([Nn]on[-][Dd]isclosure)|(NON[-]DISCLOSURE)*”. Semua dokumen kontrak yang diekstrak dari EDGAR memiliki format HTML dan teks murni.

3.6 Teknik Keabsahan Data

Keabsahan pada data CNLI dapat dilihat dari adanya leaderboard model dan penelitian-penelitian terdahulu yang telah dilakukan. Leaderboard dibuat untuk melacak progres perkembangan model-model NLI yang telah dapat menyelesaikan tugas NLI menggunakan data CNLI, dan dapat dilihat pada halaman web paperswithcode.com. Beberapa model yang telah diteliti menggunakan data CNLI antara lain, Span NLI BERT (Koreeda & Manning, 2021), UL2 (Tay dkk., 2022), CoLT5 (Ainslie dkk., 2023), dan SCROLLS (Shaham dkk., 2022). Dari penelitian-penelitian pada model-model tersebut, dapat disimpulkan bahwa CNLI merupakan dataset yang didesain khusus untuk tugas NLI dan layak digunakan dalam penelitian yang bertujuan untuk membuat model NLP dengan fokus inferensi bahasa (*Language Inference*).

3.7 Teknik Analisis Data

3.7.1 Analisis Metode

Untuk menganalisis metode yang digunakan, peneliti menggunakan beberapa hyperparameter berbeda untuk melakukan pre-training terhadap data CNLI. Hal ini dilakukan untuk menemukan model yang paling optimal dalam konteks tak hanya akurasi tetapi juga efisiensi memori dan waktu. Contoh tabel analisis yang akan digunakan dapat dilihat pada Tabel 3.3. Beberapa hyperparameter seperti batch size dan hidden groups akan diubah dan disesuaikan untuk menemukan yang paling optimal. *Hidden groups* berfungsi untuk mendefinisikan berapa grup layer yang akan saling berbagi parameter. F1 dihitung untuk 2 (dua) label, yaitu entail dan contradiction.

Tabel 3. 4 Contoh Hyperparameter untuk model ALBERT

<i>Pre-trained</i>	<i>Batch Size</i>	<i>Epoch</i>	<i>Learning Rate</i>	<i>Hidden Groups</i>	<i>Early Stop</i>
<i>true</i>	20	20	2e-6	3	<i>true</i>
<i>true</i>	16	20	2e-6	3	<i>true</i>
<i>true</i>	8	20	2e-6	3	<i>true</i>

3.7.2 Analisis Hasil

Karena tugas NLI pada dataset CNLI yang dikerjakan dapat dikategorikan sebagai *multi-class classification*, maka metrik yang dipakai untuk mengevaluasi hasil penelitian adalah F1, Precision, Recall, dan Akurasi. Seperti yang terlihat pada Persamaan 3.1, skor F1 mikro dihitung dengan menggunakan total *True Positive*, *False Positive*, dan *False Negative* tanpa mempedulikan tiap kelas. Skor F1 mikro memberikan tiap kelas bobot yang sama, termasuk yang dari kelas mayoritas (Allwright, 2022). Hal ini dapat memberikan hasil yang kurang baik terutama pada data yang kelasnya tidak seimbang.

$$F1_{micro} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3.1)$$

$$F1_{macro} = \frac{F1_1 + F1_2 \dots + F1_n}{n} \quad (3.2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3.3)$$

$$Recall = \frac{TP}{TP+FN} \quad (3.4)$$

$n = \text{banyak kelas}$

$TP = \text{True Positive}$

$FP = \text{False Positive}$

$FN = \text{False Negative}$

Maka dari itu, peneliti memutuskan untuk menggunakan F1 makro pula. Seperti yang terlihat pada Persamaan 3.2, skor F1 makro didapat dengan cara menghitung skor F1 tiap kelas secara terpisah, lalu diaplikasikan rata-rata. Hal ini bertujuan untuk memberikan bobot yang sesuai pada tiap kelas (Allwright, 2022). Karenanya, skor F1 makro cenderung lebih baik dalam menghitung akurasi pada data yang kelasnya tidak seimbang, seperti CNLI.

BAB 4

HASIL PENELITIAN DAN PEMBAHASAN

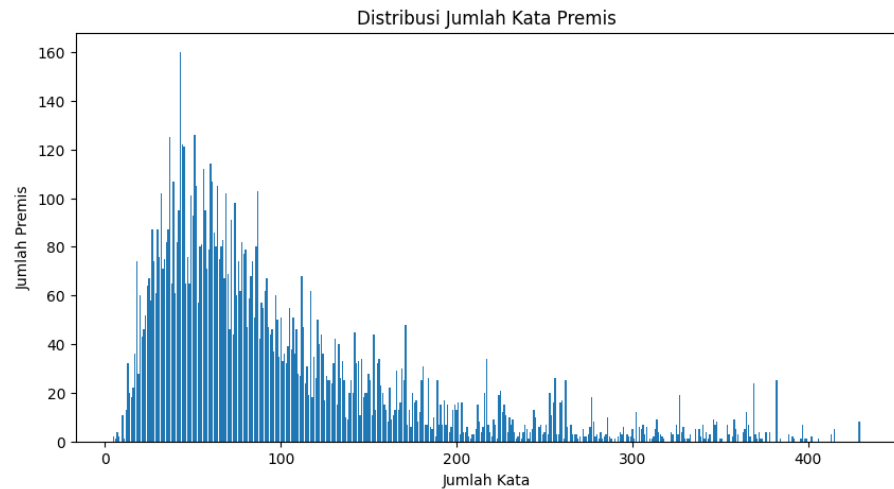
4.1 Hasil Penelitian

Penelitian yang telah dilakukan dengan menggunakan model ALBERT-base dengan tujuan untuk menentukan keterikatan tekstual antara hipotesis dan premis dalam dataset yang berisi surat kontrak NDA menunjukkan hasil yang melebihi ekspektasi peneliti. Proses pengolahan data Contract NLI hingga evaluasi metrik skor F1 dan akurasi dilakukan beberapa kali dengan *hyperparameter* berbeda untuk memastikan konsistensi hasil dan aspek *repeatability*, atau kemungkinan dimana penelitian dapat diduplikat dengan *setup* yang sama dan dijamin menghasilkan hasil yang sama pula.

4.1.1 Hasil Pengolahan Data

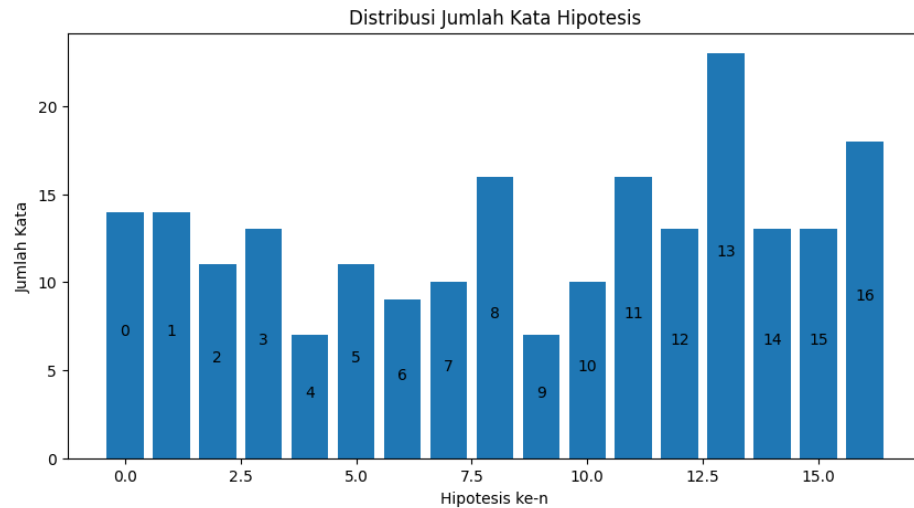
4.1.1.1 Hasil Persebaran Jumlah Kata CNLI

Dataset CNLI (*Contract NLI*) yang dibuat pada 2021 mengandung 607 premis berupa file dokumen surat kontrak dan 17 hipotesis. Setiap file dokumen dengan format HTML, TXT, dan PDF memiliki ukuran bervariasi dan dapat terdiri dari 2 atau lebih halaman. Setelah melewati tahap *preprocessing* yaitu pembersihan teks premis dari surat kontrak dan hanya diambil bagian pentingnya saja, ukuran teks premis mengecil secara drastis. Pada Gambar 4.1 terlihat persebaran jumlah kata pada premis dataset menunjukkan bahwa kebanyakan premis yang digunakan untuk *training* memiliki banyak kata tidak lebih dari 120. Meskipun begitu, model Albert tetap dapat memahami data dengan baik karena premis masih mengandung informasi penting dari file kontrak asal.



Gambar 4. 1 Grafik distribusi kata dari teks surat kontrak CNLI (premis)

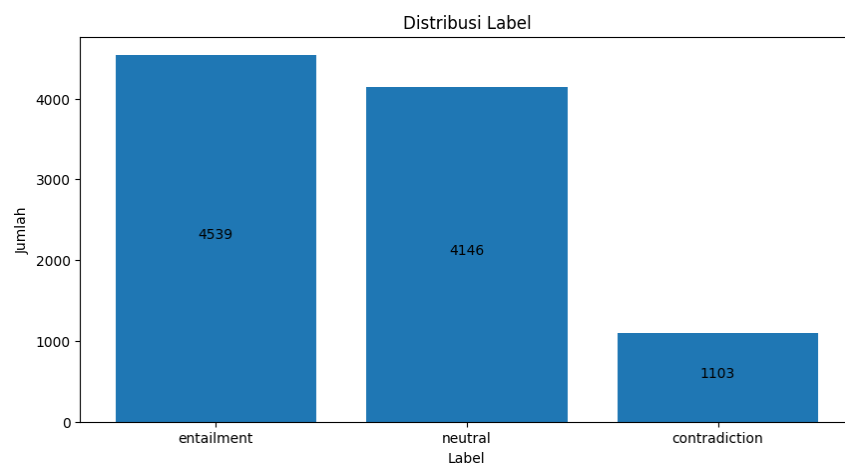
Hipotesis dari data CNLI telah didesain sedemikian rupa untuk menguji *Deep Learning Model* dalam melakukan tugas inferensi, terutama keterikatan tekstual. Hipotesis berjumlah 17 ini memiliki jumlah kata mulai kurang dari 10 hingga lebih dari 20, seperti yang terlihat pada Gambar 4.2. Untuk menentukan keterikatan tekstual, model Albert perlu memahami konteks dari premis dan hipotesis secara mendalam. Hal ini diperlukan karena hipotesis telah dirancang untuk mencegah model dalam menebak keterikatan tekstual hanya dengan mengenali struktur kalimat dan menggunakan kata berulang.



Gambar 4. 2 Grafik distribusi kata dalam hipotesis dari dataset CNLI

4.1.1.2 Hasil Persebaran Label Data

Dari 607 premis dan 17 hipotesis, total data yang berakhir digunakan dalam proses training menjadi 9.788 pasang data. Persebaran label atau kelas dari data tersebut dapat dikatakan tidak seimbang, seperti yang terlihat pada Gambar 4.3. Kelas *Entailment* menempati 46% dari total keseluruhan data. Pada kasus ini, proses evaluasi memerlukan metrik khusus yang dapat mengukur performa model yang dilatih dengan data yang kelasnya tidak seimbang, seperti F1 Makro.



Gambar 4. 3 Distribusi data CNLI berdasarkan label entail, contradict, dan neutral

Ketiga kelas atau label yang digunakan untuk menentukan tingkat keterikatan tekstual dari data CNLI adalah “*Entail*”, “*Neutral*”, dan “*Contradict*”. Kelas tersebut menunjukkan nilai kebenaran dari satu hipotesis saat diberikan sebuah premis berupa dokumen surat kontrak NDA. Hubungan premis-hipotesis ini bersifat satu arah, dimana jika premis P entail hipotesis H, maka sebaliknya belum tentu hipotesis H entail premis P. Berikut penjelasan dari kelas dalam tugas keterikatan tekstual CNLI.

1) *Entail*

Pasangan hipotesis dan premis dapat dikategorikan sebagai *entail* jika saat diberikan premis berupa teks T lalu diberikan hipotesis H, model dapat menentukan bahwa hipotesis itu bersifat benar. Contoh sederhana dari *entailment* adalah premis “Budi dan Nina berada di lokasi kecelakaan mobil saat itu terjadi” dan hipotesis “Beberapa orang melihat kecelakaan mobil itu”.

2) *Contradict*

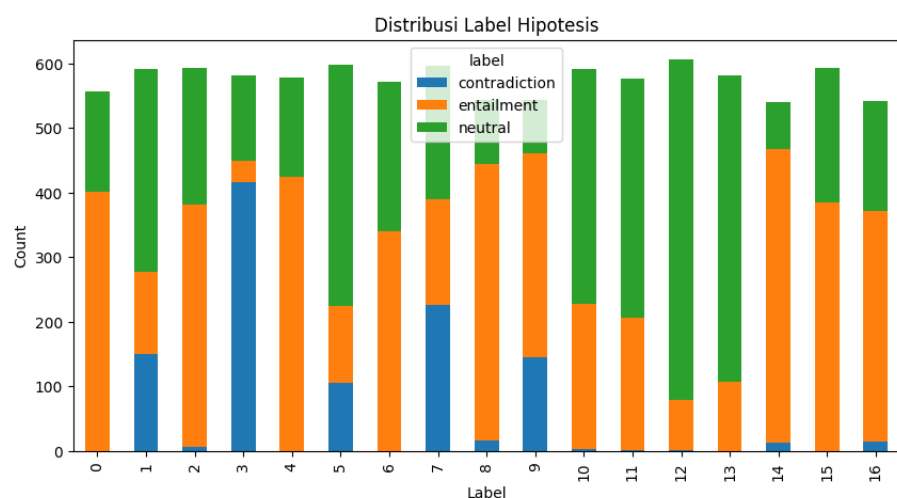
Hipotesis yang diterima dikategorikan berkontradiksi (*contradict*) dengan premis teks yang ada jika hipotesis tersebut tidak dapat diterima kebenarannya. Contoh sederhana dari *contradiction* adalah premis “dua anjing bermain di taman dengan seorang kakek tua” dan hipotesis “Hanya ada satu hewan yang bermain di taman saat itu”.

3) *Neutral*

Neutral atau terkadang disebut *Not Mentioned* merupakan label atau kelas yang diberikan jika teks premis dan hipotesis tidak berhubungan. Premis tidak menyediakan cukup informasi bagi model untuk menentukan nilai kebenaran hipotesis yang diberikan. Contoh sederhananya adalah premis “aku bermain basket

dengan anak kecil itu” dan hipotesis “anak kecil menyukai es krim”. Premis tidak menyediakan cukup informasi tentang sentimen anak kecil terhadap es krim.

Serupa dengan premis teks surat kontrak CNLI, setelah data melalui tahap preprocessing, jumlah hipotesis relatif terhadap label pun tidak seimbang persebarannya. Seperti yang terlihat pada Gambar 4.4, seluruh hipotesis yang berjumlah 17 memiliki persentasi label yang sangat bervariasi. Hipotesis dengan indeks 0 tidak memiliki hipotesis pada kelas contradiction, sedangkan hipotesis dengan indeks 3 memiliki banyak kelas contradiction dan memiliki sedikit sekali kelas entailment. Contoh lain adalah hipotesis dengan indeks 4, 10, 11, 12, 13, dan 15 yang memiliki sedikit hingga tanpa kelas contradiction. Hal ini sangat mempengaruhi proses training model, karena membuat sampel yang digunakan untuk memahami kelas menjadi terlalu sedikit.

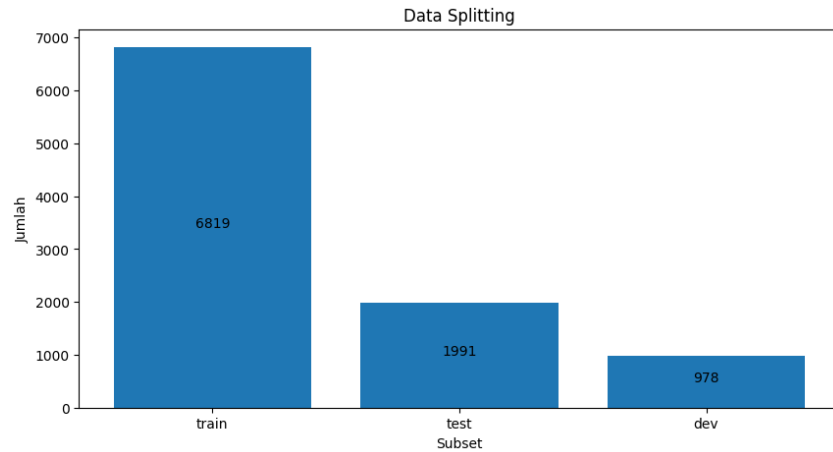


Gambar 4. 4 Grafik persebaran label (kelas) dalam tiap hipotesis

4.1.1.3 Hasil Splitting Data

Pembagian kuantitas data dalam proses splitting sudah mengikuti ketentuan dari dalam dataset. Pada Gambar 4.5 terlihat subset training memiliki persentasi 70%, subset testing memiliki

persentasi 20%, dan subset dev atau validasi memiliki persentasi 10%. Jumlah subset training jauh lebih banyak karena bertujuan untuk memaksimalkan proses model dalam memahami data.



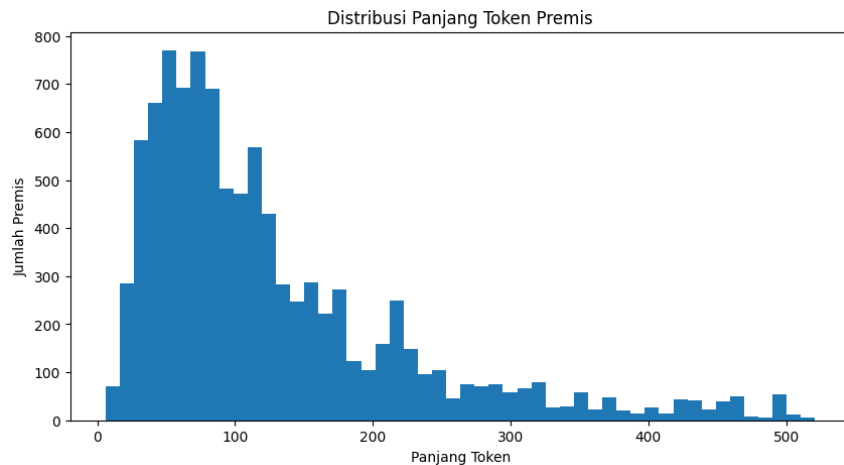
Gambar 4. 5 Grafik persebaran data pada subset *training*, *validation* (dev), dan *testing*

4.1.1.4 Hasil Tokenisasi Data

Tokinezing adalah proses dimana input teks dipecah menjadi potongan kecil yang disebut token. Pemecahan ini dapat dilakukan berdasarkan kata, sub-kata, atau huruf. Proses ini bertujuan supaya model dapat lebih mudah dalam memahami teks input, dengan memecah teks menjadi potongan kecil membuat ekstraksi fitur, pengambilan informasi, dan menejemen vokabulari menjadi lebih efektif. Pada penelitian ini, tokenizer yang digunakan adalah AlbertTokenizer. Tokenizer ini didasarkan pada sentencepiece library, dan menggunakan metode subword unit (*byte-pair-encoding* atau unigram language model) untuk memecah teks dan menciptakan token.

Karena menggunakan metode subword dalam tokenisasi teks, hasil token yang dibuat dari data CNLI pun memiliki jumlah yang sedikit lebih banyak dari total jumlah kata teks input. Pada Gambar 4.6 merupakan contoh distribusi total hasil tokenisasi teks premis dari data CNLI. Sekilas terlihat bentuk kumpulan bar mirip seperti

distribusi panjang premis pada Gambar 4.1, perbedaannya adalah distribusi token premis sedikit bergeser ke kanan melewati titik 500. Hal ini menunjukkan bahwa rata-rata panjang token lebih panjang dari rata-rata panjang teks input premis data CNLI.



Gambar 4. 6 Grafik distribusi panjang token pada tiap teks surat kontrak (premis) dataset CNLI

Detail tentang hasil tokenisasi data CNLI premis dan hipotesis dideskripsikan pada Tabel 4.1. Token premis memiliki rata-rata panjang 126, dimana panjang minimum adalah 6 token dan panjang maksimum adalah 521 token. *Standard Deviation* (Std) menunjukkan angka yang cukup tinggi pada 98, menunjukkan bahwa panjang hasil tokenisasi premis sangat bervariasi dan menyebar sepanjang axis x. Sebaliknya, token pada hipotesis memiliki nilai *Standard Deviation* yang cenderung kecil, karena panjang teks yang dimiliki didesain untuk tidak jauh berbeda. Hipotesis memiliki panjang minimum 8 dan maksimum 28 token, dengan rata-rata memiliki kurang lebih 15 buah token.

Tabel 4. 1 Detail hasil tokenisasi premis dan hipotesis dataset CNLI

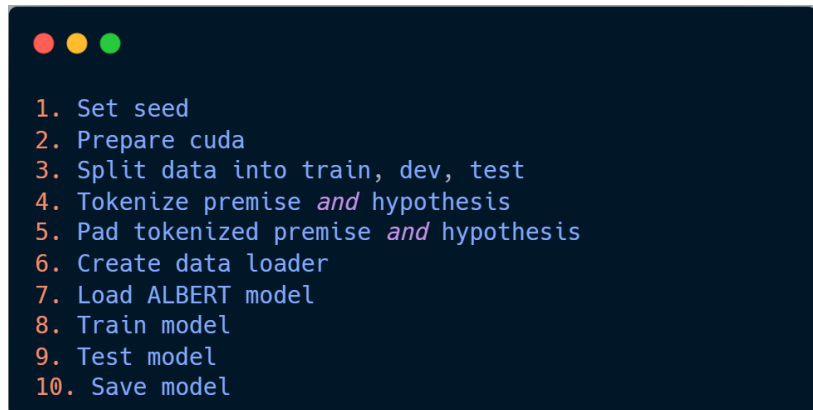
	Premis	Hipotesis
Count	9788	9788
Mean	126,725276	15,587148

Std	98,125668	5,262601
Min	6	8
25%	59	12
50%	96	15
75%	162	17
Max	521	28

4.1.2 Hasil Implementasi ALBERT

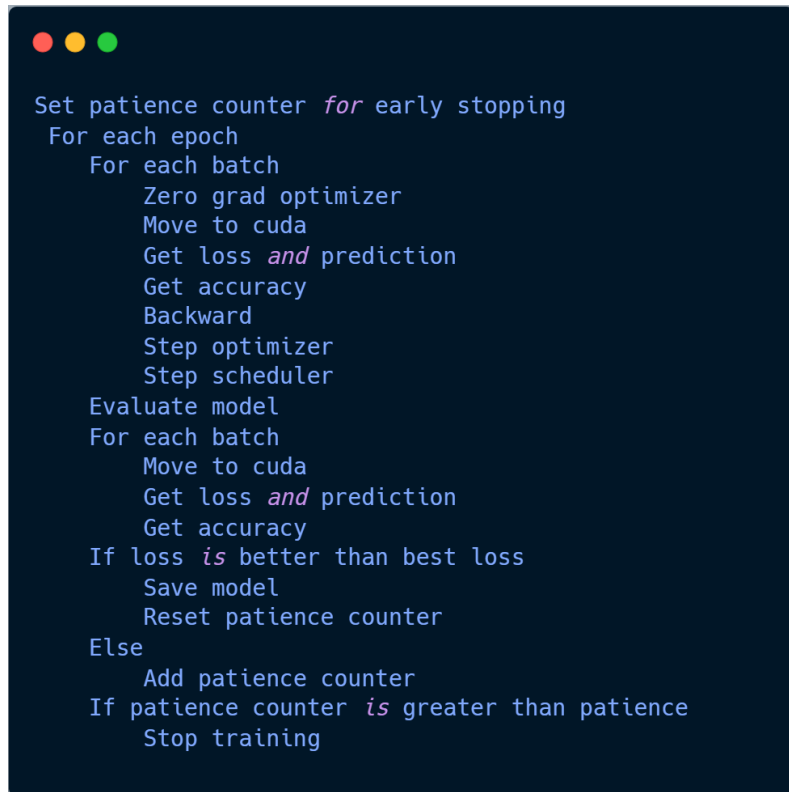
4.1.2.1 Hasil Pseudocode

Sebelum melakukan implementasi model ALBERT, perlu adanya struktur kode atau pseudocode untuk mempermudah dalam pengaplikasian dan pengevaluasian kode. *Library* yang digunakan selama proses penelitian adalah PyTorch. *Library* yang dikembangkan oleh Meta AI. Seperti yang dideskripsikan pada Gambar 4.7, poin 1 menunjukkan langkah pertama yaitu seeding. Seeding berfungsi untuk memastikan konsistensi dalam hasil eksperimen. Selanjutnya adalah poin 2 yang bertujuan untuk memeriksa apakah hardware yang digunakan memiliki GPU (*Graphical Processing Unit*). Setelah itu dilakukan proses pemecahan dataset CNLI menjadi subset train, dev, dan test. Lalu dilakukan proses tokenisasi teks premis dan hipotesis dengan menggunakan ALBERT Tokenizer. Selanjutnya diimplementasikan proses padding pada data dengan tujuan supaya data input memiliki panjang yang sama. Setelah itu data dimasukkan ke dalam loader sesuai dengan ukuran *batch size*. Terakhir, data dimasukkan ke dalam model ALBERT, untuk selanjutnya dilakukan proses training. Detail pada proses training dijelaskan pada gambar 4.8.



Gambar 4. 7 Alur proses *pre-training* model dengan data CNLI

Pada proses training seperti yang terlihat pada Gambar 4.9, metode early stopping digunakan untuk mencegah terjadinya *overfitting*. *Overfitting* terjadi ketika model mempelajari data training terlalu baik, dan performanya menjadi buruk ketika menemui sampel data yang belum pernah ditemui saat training. Selanjutnya dilakukan perulangan berdasarkan jumlah epoch dan batch size. Pada *forward training*, data dipindahkan kedalam cuda dan dihitung nilai loss dan hasil prediksinya. Setelahnya, akan dilakukan backward training, yang bertujuan untuk mengubah nilai parameter pada model dan *optimizer* supaya mendekati titik optimal. Setelah dilakukan training, model akan menjalankan proses evaluasi. Proses ini sama seperti proses training, hanya saja menggunakan data pada subset *dev/validation*, dan tidak melakukan backward untuk mengubah nilai parameter. Terakhir, dilakukan pemeriksaan nilai *loss*. Jika 2 (dua) nilai *loss* yang muncul lebih besar dari nilai *loss* sebelumnya, proses training akan dihentikan.



Gambar 4. 8 Detail alur proses *training* model ALBERT-base

4.1.2.2 Hasil Tuning Hyperparameter

Setelah dilakukan eksperimen dan proses training berulang kali, peneliti menemukan *hyperparameter* yang dikatakan optimal untuk model ALBERT dalam melakukan tugas keterikatan tekstual dengan data CNLI. Seperti yang terlihat pada Tabel 4.2, *Batch Size* (BS) yang digunakan adalah 8, 16, 20. BS merupakan salah satu hyperparameter yang sangat penting. Semakin besar ukurannya, semakin cepat proses training dan semakin kecil ukurannya, semakin akurat hasil prediksinya. *Learning Rate* (LR) yang digunakan adalah $2e-5$ dan $2e-6$. *Weight Decay* (WD) atau biasa disebut L2 Regularization yang digunakan adalah 0 dan 0,01, tergantung jenis layer pada model. WD berfungsi untuk mencegah overfitting, dengan cara memberikan penalti pada loss function.

Tabel 4. 2 Hyperparameter yang dipakai dalam proses training

<i>Hyperparameter</i>	Nilai		
<i>Batch Size</i>	8	16	20
<i>Learning Rate</i>	2e-5		2e-6
<i>Weight Decay</i>	0		0,01
<i>Warmup Steps</i>	0		100
<i>Epochs</i>	10		
<i>Layer Groups</i>	0		3
<i>Early Stop</i>	<i>True</i>		

Warmup Step (WS) berfungsi untuk menjaga proses training tetap stabil dengan cara meningkatkan nilai LR dari sangat kecil menuju nilai LR yang telah ditetapkan. WS yang dipakai adalah 100 steps. *Layer Groups* atau *Hidden Group* berfungsi untuk mendefinisikan berapa grup layer yang akan saling berbagi parameter. Nilai *Layer Group* yang peneliti gunakan adalah 0 dan 3.

4.1.2.3 Hasil Pretraining Model

Pretraining adalah proses melatih seluruh layer model ALBERT dengan data baru, yaitu CNLI. Model ALBERT yang digunakan merupakan “albert-base-v2” yang diambil melalui <https://huggingface.co>. Pada pengimplementasiannya, peneliti menerapkan 2 (dua) jenis model ALBERT-base, yaitu dengan konfigurasi manual dan konfigurasi default, seperti yang terlihat pada Gambar 4.9. Model dengan konfigurasi manual digunakan ketika eksperimen menggunakan *Layer Groups* (LG) lebih dari 1, sedangkan model dengan konfigurasi default digunakan ketika LG hanya bernilai 1.

```

from transformers import AlbertForSequenceClassification, AlbertConfig

if NUM_HIDDEN_GROUPS > 1:
    # Initialize the configuration
    config = AlbertConfig(
        num_hidden_layers=12, # Total number of layers
        num_attention_heads=12,
        hidden_size=768,
        intermediate_size=3072,
        num_labels=3,
        num_hidden_groups=NUM_HIDDEN_GROUPS, # Number of layer groups
    )
    model = AlbertForSequenceClassification.from_pretrained("albert-base-v2", config=config)
else:
    model = AlbertForSequenceClassification.from_pretrained("albert-base-v2", num_labels=3)
model.to(device)

```

Gambar 4. 9 Source code penggunaan model ALBERT-base dari *huggingface.co*

Model ALBERT-base dengan konfigurasi default yang digunakan dalam eksperimen memiliki banyak layer yang dapat dikelompokkan menjadi 4 kelompok, yaitu layer *embedding*, layer *encoder*, layer *pooler*, dan layer *classifier*. Seperti yang terlihat pada Tabel 4.3, pada kelompok *embedding*, terdapat layer *word embedding* yang berfungsi untuk menerima input *embedding* dari data *loader*, *position embedding* yang berfungsi untuk mlacak posisi token dalam teks, dan *token type embedding* yang berfungsi untuk menandai akhir dari sebuah *input sequence*. Sedangkan layer Norm pada kelompok *embedding* berfungsi untuk menormalisasikan tiap fitur pada data dengan tujuan untuk menstabilkan proses *training*.

Tabel 4. 3 Layer model ALBERT-base dengan konfigurasi 1 (satu) *layer group* dan *classifier head*

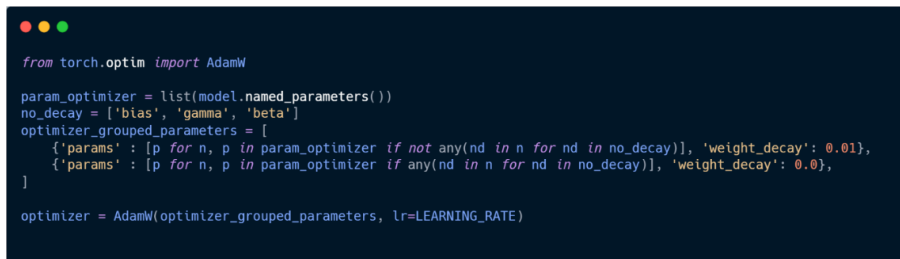
No	Layer	Dimensi
Embedding		
1	word_embeddings.weight	[30000, 128]
2	position_embeddings.weight	[512, 128]
3	token_type_embeddings.weight	[2, 128]
4	LayerNorm.weight	[128]
5	LayerNorm.bias	[128]
Encoder (Layer Groups Index 0)		

6	embedding_hidden_mapping_in.weight	[768, 128]
7	embedding_hidden_mapping_in.bias	[768]
8	albert_layers.0.full_layer_layer_norm.weight	[768]
9	albert_layers.0.full_layer_layer_norm.bias	[768]
10	albert_layers.0.attention.query.weight	[768, 768]
11	albert_layers.0.attention.query.bias	[768]
12	albert_layers.0.attention.key.weight	[768, 768]
13	albert_layers.0.attention.key.bias	[768]
14	albert_layers.0.attention.value.weight	[768, 768]
15	albert_layers.0.attention.value.bias	[768]
16	albert_layers.0.attention.dense.weight	[768, 768]
17	albert_layers.0.attention.dense.bias	[768]
18	albert_layers.0.attention.LayerNorm.weight	[768]
19	albert_layers.0.attention.LayerNorm.bias	[768]
20	albert_layers.0.ffn.weight	[3072, 768]
21	albert_layers.0.ffn.bias	[3072]
22	albert_layers.0.ffn_output.weight	[768, 3072]
23	albert_layers.0.ffn_output.bias	[768]
Pooler		
24	albert.pooler.weight	[768, 768]
25	albert.pooler.bias	[768]
Classifier		
26	classifier.weight	[3, 768]
27	classifier.bias	[3]

Kelompok layer *encoding* pada arsitektur model ALBERT-base memiliki fungsi untuk memproses *input sequence* dan menciptakan representasi angka yang menangkap makna dan konteks dari sequence tersebut. Pada Tabel 4.3 *layer encoding* mengandung indeks 0 (no1) pada nama layer, karena ini merupakan arsitektur dari model ALBERT-base dengan 1 (satu) *layer group*

yang saling berbagi parameter. Sedangkan layer pooler pada model berfungsi untuk mengubah output dari layer sebelumnya menjadi vektor angka dengan ukuran tertentu (dalam kasus ini 768). Dengan mengubah dimensi output menjadi lebih padat, memungkinkan model untuk memanfaatkan hasil output sesuai dengan tugas yang dikerjakan (dalam kasus ini NLI).

Selanjutnya dalam proses pretraining adalah mendefinisikan *optimizer*. *Optimizer* memiliki fungsi untuk mengubah parameter model dengan tujuan untuk meminimalkan nilai loss supaya hasil prediksi model semakin mendekati titik target yang dipredikis. Proses ini disebut optimisasi. Seperti yang terlihat pada Gambar 4.10, optimizer yang dipakai dalam penelitian adalah AdamW. AdamW merupakan variasi dari Adam (*Adaptive Moment Estimation*) yang memisahkan weight decay dari proses optimization step. AdamW berkerja lebih baik dalam menangani weight decay dan overfitting saat proses training. Pada penelitian ini, optimizer AdamW menggunakan 2 (dua) *weight decay*, yaitu 0 dan 0,01. *Weight decay* 0,01 diaplikasikan pada semua layer kecuali layer dengan tipe bias, gamma, dan beta.



```
from torch.optim import AdamW

param_optimizer = list(model.named_parameters())
no_decay = ['bias', 'gamma', 'beta']
optimizer_grouped_parameters = [
    {'params': [p for n, p in param_optimizer if not any(nd in n for nd in no_decay)], 'weight_decay': 0.01},
    {'params': [p for n, p in param_optimizer if any(nd in n for nd in no_decay)], 'weight_decay': 0.0},
]

optimizer = AdamW(optimizer_grouped_parameters, lr=LEARNING_RATE)
```

Gambar 4. 10 *Source code* penggunaan optimizer AdamW

Scheduler atau *learning rate scheduler* berfungsi untuk menyesuaikan *learning rate* secara dinamis selama proses training. Penyesuaian ini dilakukan dengan cara mengubah nilai *learning rate* sesuai dengan strategi yang telah didefinisikan pada *scheduler* sebelumnya. Pada penelitian ini, *scheduler* yang dipakai adalah LambdaR. LambdaR memungkinkan untuk mendefinisikan fungsi

baru yang dapat disesuaikan dengan kebutuhan secara manual. Seperti yang terlihat pada Gambar 4.11, strategi scheduler yang digunakan pada penelitian ini adalah dengan meningkatkan nilai *learning rate* secara perlahan setiap step sebanyak nilai WARMUP_STEPS. Saat training step telah melebihi nilai WARMUP_STEPS, nilai learning rate akan menurun secara perlahan sebanyak sisa TOTAL_TRAIN_STEP – WARMUP_STEPS. Strategi ini digunakan supaya model tidak menjauh dari titik optimal karena learning rate yang terlalu besar.



```

TOTAL_TRAIN_STEP = len(train_loader)

def warmup(current_step: int):
    if current_step < WARMUP_STEPS:
        return float((current_step + 1) / WARMUP_STEPS)
    else:
        return max(0.0, float(TOTAL_TRAIN_STEP - current_step) / float(max(1, TOTAL_TRAIN_STEP - WARMUP_STEPS)))

lr_scheduler = torch.optim.lr_scheduler.LambdaLR(optimizer, lr_lambda=warmup)

```

Gambar 4. 11 *Source code* penggunaan *early stopping* dengan *lambdaLR* pada optimizer

4.1.2.4 Hasil Evaluasi Keterikatan Tekstual

Proses *pretraining* model ALBERT-base dengan data CNLI dibagi ke dalam 4 (empat) kategori, yaitu tanpa *warmup step* dan satu *layer group*, tanpa *warmup step* dan tiga *layer group*, dengan *warmup step* dan satu *layer group*, dan dengan *warmup step* dan tiga *layer group*.

4.1.2.4.1 Hasil Satu Layer Group

Pada Tabel 4.4 terlihat bahwa kelompok dengan *warmup step* bernilai 0 memiliki perbedaan signifikan dengan kelompok yang menggunakan 100 *warmup step*. Tiap eksperimen pada tabel dilakukan sebanyak 2 kali untuk memastikan konsistensi. Semua proses training menggunakan algoritma *early stop*. Eksperimen dengan BS 20 dan WS 0 menghasilkan nilai akurasi pada data validasi dan skor F1-macro terbesar dibanding eksperimen dengan hyperparameter lain, namun hasil akurasi pada data testing terbesar dicapai dengan BS 8 dan WS 0. Pada Tabel 4.4 juga terlihat pola

peningkatan akurasi yang mengikuti menurunnya nilai BS. Hal ini menunjukkan bahwa model dapat mempelajari data dengan baik dan memiliki potensi menghasilkan akurasi lebih tinggi saat menggunakan BS sebesar 8.

Tabel 4. 4 Hasil eksperimen *pre-training* model dengan satu *hiddnen layer group*

WS	BS	HG	Acc (%)		F1mic			F1 mac	ES
			Val	Test	entail	Contra	neutral		
0	20	1	84,7	84,9	86,2	75,7	85,5	82,5	T
0	16	1	84,1	83,9	84,8	75,3	84,7	81,6	T
0	8	1	84,6	85,0	86,5	74,8	85,8	82,4	T
100	20	1	82,2	80,8	82,0	65,7	82,4	76,7	T
100	16	1	78,6	79,1	80,6	66,3	80,0	75,6	T
100	8	1	82,1	82,3	84,0	70,3	83,3	79,2	T

WS (Warmup Step), BS (Batch Size), HG (Hidden Groups), Fmic (F1 micro), F1mac (F1 macro), ES (Early Stop), Acc (Accuracy)

4.1.2.4.2 Hasil Tiga Layer Group

Tabel 4.5 menunjukkan hasil eksperimen dari model ALBERT-base menggunakan 3 *layer groups*. Tiga *layer groups* dapat diartikan bahwa blok attention dalam model dibagi menjadi 3 grup dengan indeks 0, 1, dan 2, dimana tiap grup akan saling berbagi parameter. Tujuannya adalah efisiensi memori. Hasil akurasi dan skor F1 pada tabel menunjukkan hasil yang berkebalikan dari hasil pada Tabel 4.4. Ukuran BS yang kecil menghasilkan akurasi yang kecil pula, sebaliknya untuk BS berukuran 20 yang menghasilkan akurasi relatif besar. Dengan menggunakan 3 *layer groups*, hasil optimal dicapai oleh model yang menggunakan WS 0 dan BS 20. Akurasi data validation mencapai 83,5 dan akurasi testing mencapai 82,6. Nilai akurasi dan

skor F1 ini masih kurang optimal jika dibandingkan dengan model yang menggunakan 1 (satu) *layer groups*.

Tabel 4. 5 Hasil eksperimen pre-training model dengan tiga *hidden layer group*

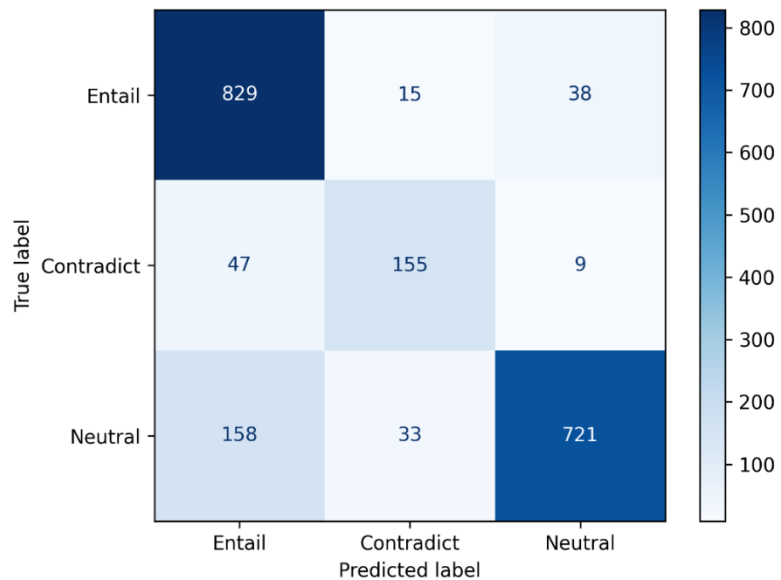
WS	BS	HG	Acc (%)		F1mic			F1 mac	ES
			Val	Test	entail	Contra	neutral		
0	20	3	83,5	82,6	84,9	72,7	82,5	80,0	T
0	16	3	83,1	83,3	85,9	74,0	82,7	80,9	T
0	8	3	81,8	81,0	84,0	70,8	80,1	78,3	T
100	20	3	79,1	78,9	81,7	68,2	77,8	75,9	T
100	16	3	78,3	77,9	81,3	66,9	76,8	75,0	T
100	8	3	42,4	39,0	53,2	7,8	23,5	28,2	T

WS (Warmup Step), BS (Batch Size), HG (Hidden Groups), Fmic (F1 micro), F1mac (F1 macro), ES (Early Stop), Acc (Accuracy)

4.1.2.4.3 Hasil Evaluasi

Setelah melakukan evaluasi dari data hasil eksperimen pretraining model ALBERT-base dengan data CNLI, didapat versi model dengan akurasi paling optimal. Model dilatih dengan menggunakan 8 *batch size*, 1 *hidden group*, *learning rate* sebesar $2e-5$, dan tanpa *warmup step*, yang dilabeli dengan nama model ALBERT-CNLI-base. Penamaan model ini didasarkan pada hyperparameter dan konfigurasi yang digunakan saat melakukan pretraining. Pada Gambar 4.12 terlihat hasil evaluasi dari model tersebut dengan menggunakan *unnormalized confusion matrix* dari data subset Testing. Gambar menunjukkan total jumlah prediksi label yang sesuai target dan jumlah prediksi yang tidak sesuai label. Axis x merupakan label yang diprediksi model dan axis y merupakan label sebenarnya. Model berhasil memprediksi 829 pasang premis-hipotesis berlabel entail, 155 pasang berlabel

contradict, dan 721 pasang berlabel neutral dengan akurat sesuai target label.



Gambar 4. 12 Hasil *Unnormalized multi-class confusion matrix* dari subset data testing

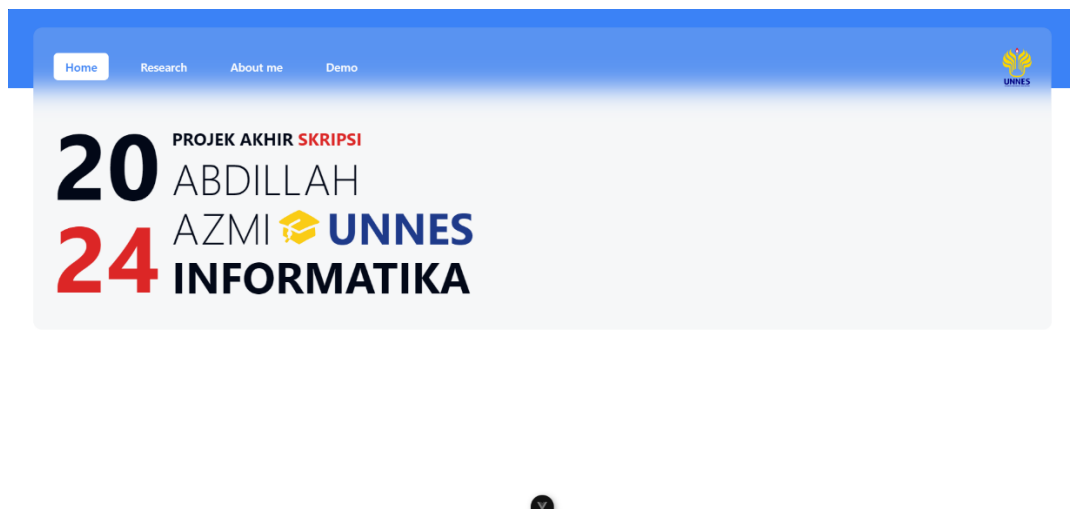
Selain menunjukkan jumlah prediksi yang sesuai target label, Gambar 4.12 juga menunjukkan *error* pada hasil prediksi model. Berikut penjelasan mengenai *error rate* yang dapat disimpulkan dari confusion matrix tersebut.

1. Pada baris 1 (label *entail*), kolom 2 dan 3 menunjukkan jumlah error (misklasifikasi) yang diprediksi model. Pada label *entail*, 53 dari total 882 data diprediksi dengan tidak akurat, hasilnya *entail* memiliki 6% *error rate*.
2. Pada baris 2, kolom 1 dan 3 menunjukkan jumlah misklasifikasi pada label *contradict*. Pada label *contradict*, 56 dari 211 prediksi merupakan error (misklasifikasi), hasilnya *contradict* memiliki 26% *error rate*. Merupakan yang paling besar diantara label lain.
3. Pada baris 3, kolom 1 dan 2 menunjukkan jumlah misklasifikasi pada label *neutral*. Pada label *neutral*, 191

dari total 912 prediksi merupakan *error*, hasilnya label *neutral* memiliki 20% *error rate*.

4.1.3 Hasil Implementasi Aplikasi Web

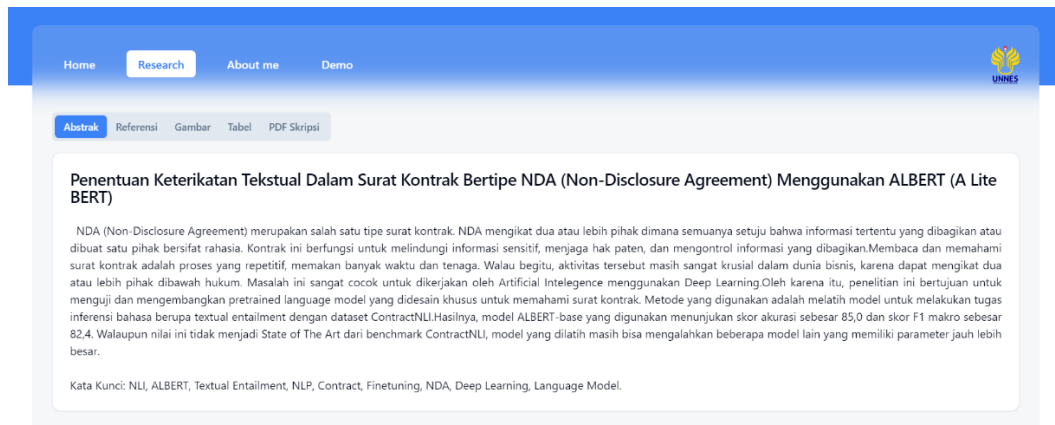
Implementasi Aplikasi yang terintegrasi dengan model dari hasil penelitian ini memanfaatkan teknologi *web framework* bernama FastAPI. *Framework* ini memungkinkan untuk membangun *backend* API dengan sangat cepat dan mudah menggunakan bahasa Python. Bagian *frontend* dari Web App memanfaatkan Vue JS dengan menggunakan Typescript. Vue JS memungkinkan untuk membuat *Single Page Application* dengan mudah. Gambar 4.13 menunjukkan halaman *Home* dari Web yang memperlihatkan tahun, nama, universitas asal, dan program studi dari penulis.



Gambar 4. 13 Halaman Home dari aplikasi web

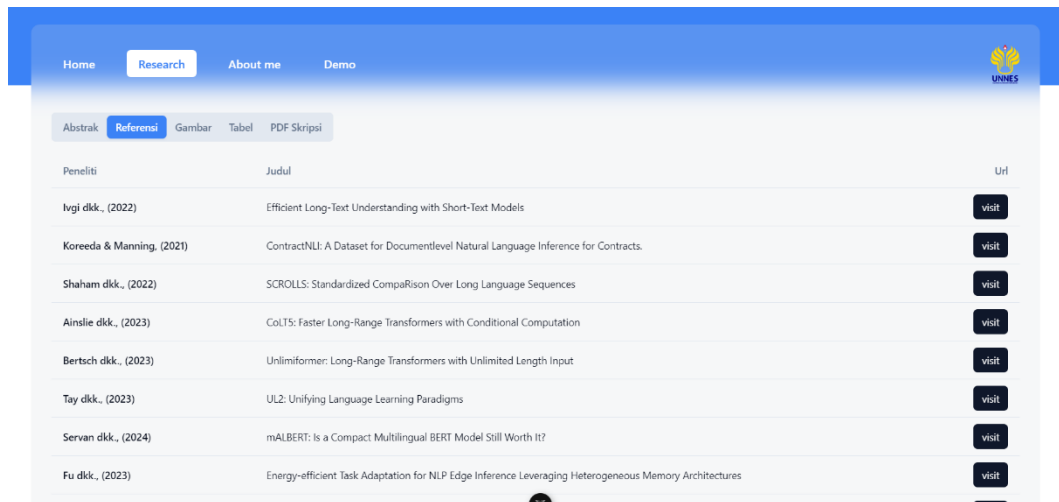
Halaman selanjutnya adalah halaman *Research*. Halaman ini menyediakan informasi lengkap tentang penelitian yang telah dilakukan. Seperti yang terlihat pada Gambar 4.14, di dalam halaman tersebut terdapat sub-menu yang menunjukkan informasi tentang abstrak, referensi, gambar, dan tabel. Selain itu terdapat juga fitur yang disediakan untuk mengunduh file skripsi dengan format PDF. Sub-menu abstrak

menunjukkan ringkasan atau *preview* dari keseluruhan penelitian, dari latar belakang hingga hasil. Halaman abstrak juga disertai dengan kata kunci yang dapat digunakan untuk mengategorikan judul penelitian ini.



Gambar 4. 14 Halaman Abstrak dari aplikasi Web

Sub-menu referensi memperlihatkan artikel-artikel yang dijadikan referensi utama dari penelitian yang dilakukan, seperti yang terlihat pada Gambar 4.15. Daftar referensi yang disertakan tidak mencakup seluruh referensi yang digunakan dalam penelitian, melainkan hanya yang utama saja. Referensi yang disertakan mengandung artikel dari penelitian-penelitian tentang pengembangan model untuk melakukan tugas inferensi dengan data CNLI dan dataset lainnya. Selain itu terdapat juga penelitian-penelitian tentang pengembangan variasi dari model ALBERT di sektor seperti biologi maupun bahasa.



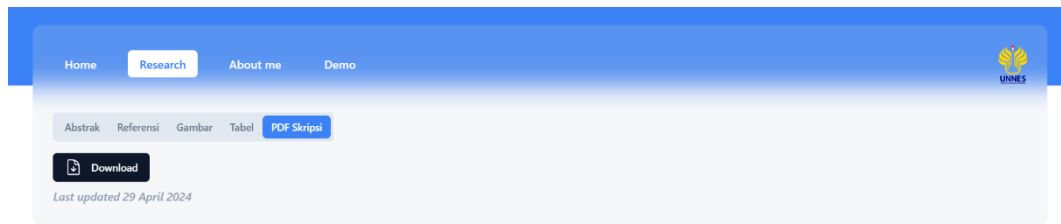
Gambar 4. 15 Halaman Referensi dari penelitian

Lalu, sub-menu gambar dan tabel menunjukkan gambar-gambar dan tabel-tabel hasil dari eksperimen yang telah dilakukan selama melakukan penelitian. Seperti yang terlihat pada gambar 4.16 dan Gambar 4.17, halaman ini menunjukkan gambar-gambar dan tabel-tabel hasil eksperimen dari penelitian yang telah dilakukan. Gambar dan tabel yang ditampilkan disertakan dengan caption yang berfungsi untuk menjelaskan isinya dengan singkat dan jelas. Gambar dan tabel dapat dipilih menggunakan tombol yang berada di kanan dan kiri, atau dapat juga menggunakan deretan angka yang berada di bawah gambar. Angka-angka ini menunjukkan indeks atau urutan dari gambar dan tabel.



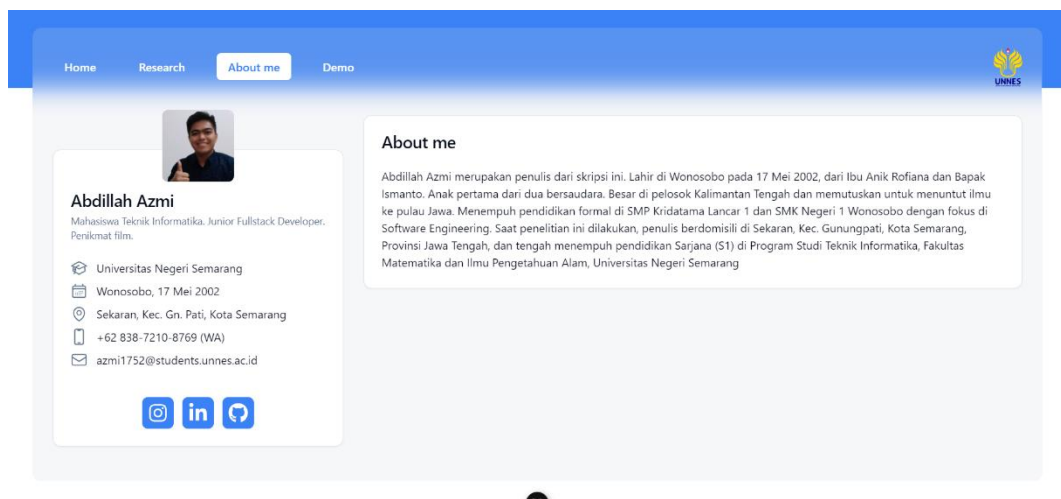
✶

Selanjutnya adalah sub-menu yang berfungsi untuk menyediakan tombol bagi pengunjung untuk mendownload file penelitian dalam format PDF. Fitur ini disediakan dengan maksud supaya jika pengunjung ingin membaca skripsi lebih detil dan memahami metode yang digunakan, serta jika ingin menduplikasi eksperimen yang dilakukan untuk membuktikan hasilnya.



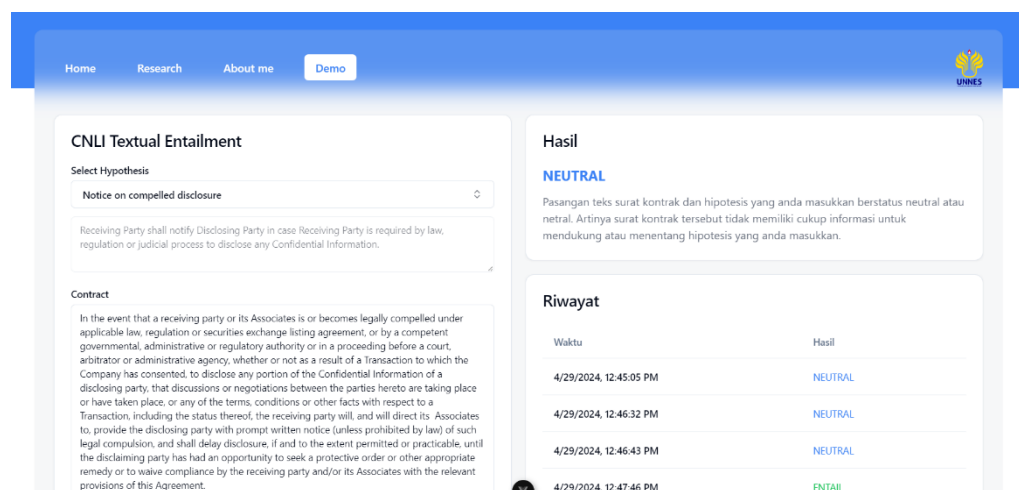
Gambar 4. 18 Halaman download file skripsi

Halaman *about me* adalah halaman yang menyediakan informasi dari penulis atau pengembang aplikasi. Pada Gambar 4.19 terlihat informasi pribadi yang ditampilkan adalah nama, asal universitas, alamat, nomor telepon, dan email. Selain itu, terdapat juga tombol yang mengarah ke sosial media. Di samping itu, halaman *about me* juga menampilkan deskripsi singkat tentang latar belakang pendidikan dan sosial pengembang.



Gambar 4. 19 Halaman About me dari aplikasi web

Terakhir adalah halaman *demo*. Halaman ini menyediakan fungsionalitas untuk memanfaatkan model AI yang diteliti dalam melakukan tugas *textual entailment*, seperti yang terlihat pada Gambar 4.20. Untuk melakukan demo, aplikasi perlu mendapatkan masukan berupa hipotesis dan teks surat kontrak. Hipotesis sebanyak 17 buah telah disediakan dan tidak memberikan kebebasan untuk mengubah. Sedangkan, surat kontrak dapat bebas ditulis oleh pengguna aplikasi. Hasil dari proses demo adalah label *neutral*, *entail*, atau *contradict*. Label ini dilengkapi dengan keterangan tambahan yang menjelaskan makna dari label tersebut. Selain itu, halaman demo juga disertai dengan fitur riwayat yang menunjukkan eksperimen-eksperimen demo yang telah dilakukan.



Gambar 4. 20 Halaman demo dari aplikasi

4.2 Pembahasan

Hasil utama dari eksperimen ditunjukkan pada Tabel 4.6. Untuk menunjukkan peningkatan performa model dalam melakukan inferensi keterikatan tekstual pada data CNLI, hasil evaluasi dibandingkan dengan model *SpanNLI-BERT* yang dibuat oleh (Koreeda & Manning, 2021). Dalam tugas inferensi keterikatan tekstual, *SpanNLI-BERT* merupakan model *multi-task* yang memanfaatkan token spesial [SPAN] yang masing-masing mewakili rentang token-token yang datang setelahnya. Model ini didesain untuk menyelesaikan 2

(dua) tugas, yaitu NLI dan *Evidence Identification*. *Loss function* yang digunakan pada tugas NLI adalah *cross-entropy*. Metriks yang digunakan untuk mengevaluasi performa NLI model adalah Akurasi, skor *F1 Contradict*, skor *F1 Entail*, dan skor *F1 macro*. Metriks ini digunakan karena data memiliki label yang tidak seimbang. Hasilnya, pada *SpanNLI BERT-base* menghasilkan akurasi sebesar 83,8 dan *SpanNLI BERT-large* menghasilkan akurasi sebesar 87,5. Model versi *base* memiliki parameter sekitar 110 juta, sedangkan versi *large* memiliki parameter sampai 340 juta. Proses *Finetuning* model *SpanNLI BERT* dilakukan sebanyak 156 kali dengan mesin ABCI (*AI Bridging Cloud Infrastructure*) yang disediakan oleh Institusi Nasional AIST (*Advanced Industrial Science and Technology*). Hasil evaluasi model *SpanNLI-BERT* pada Tabel 4.6 memperlihatkan hasil akurasi dalam tugas NLI dengan data CNLI, diikuti dengan skor *F1* dari label *contradict* dan skor *F1* dari label *entail*. *SpanNLI-BERT-large* memiliki akurasi jauh lebih besar karena menggunakan *BERT-large* yang memiliki parameter lebih besar dan memiliki kemampuan memahami konteks lebih tinggi dari *BERT-base*.

Tabel 4. 6 Hasil utama eksperimen dengan nilai akurasi terbaik

Peneliti	Model	Acc	F1(C)	F1(E)
(Koreeda & Manning, 2021)	SpanNLI-BERT-base	83,8	28,7	76,5
	ALBERT-CNLI-base	85,0	74,8	86,5

Tabel 4.7 menunjukkan performa beberapa model yang dilatih dengan data CNLI untuk melakukan tugas inferensi keterikatan tekstual. Metriks yang digunakan untuk mengevaluasi model adalah EM (Exact Match). EM merupakan metriks yang memberikan nilai kemiripan saat dua atau lebih string dibandingkan. EM menuntut hasil prediksi model dan target yang diprediksi untuk benar-benar sama, jadi hasil perbandingan tiap baris prediksi-target hanya berupa “benar” dan “salah”, tidak ada konsep “hampir benar”. EM sering kali digunakan untuk membandingkan data sekuen berupa string dan menuntut kecocokan. Pada kasus ini adalah label yang diprediksi model dan label target.

Tabel 4. 7 Hasil EM (*Exact Match*) dari model

Peneliti	Model	EM (%)
(Tay dkk., 2022)	<i>UL20B</i>	88,7
(Ainslie dkk., 2023)	<i>CoLT5-Large</i>	88,7
(Ivgi dkk., 2023)	<i>SLED (BART-large)</i>	87,3
(Bertsch dkk., 2023)	<i>Unlimiformer (BART-base)</i>	77,7
(Shaham dkk., 2022)	<i>SCROLLS (BART-base)</i>	77,4
	ALBERT-CNLI-base	85,04

Hasil evaluasi EM yang ditunjukkan oleh Tabel 4.7 memperlihatkan bahwa model UL20B yang memiliki 20 miliar parameter dan CoLT5-Large yang memiliki kurang lebih 770 juta parameter memiliki nilai EM yang sama, yaitu 88,7% pada tugas inferensi keterikatan tekstual dengan data CNLI. Model BART-large dengan kurang lebih 406 juta parameter yang menggunakan metode SLED (Sliding Encoder Decoder) menghasilkan EM sebesar 87,3. Ketiga model tersebut memiliki nilai EM yang relatif lebih besar dari model yang sedang diteliti. Sedangkan untuk model BART-base dengan jumlah parameter kurang lebih 140 juta yang menggunakan Unlimiformer dan SCROLLS menghasilkan nilai EM dibawah 80%. Hal ini menunjukkan bahwa model ALBERT-CNLI-base sudah dapat menandingi model dengan ratusan juta parameter dalam melakukan tugas inferensi keterikatan tekstual dengan data CNLI, namun masih kesulitan dalam menandingi model yang didasarkan dari T5-large dan BART-large.

Shaham dkk., (2022) mengenalkan benchmark bernama SCROLLS (*Standardized Comparison Over Long Language Sequences*) yang didesain untuk menantang model-model dalam menyelesaikan tugas yang membutuhkan pemahaman dan penalaran terhadap teks panjang. Tugas-tugas di dalam SCROLLS diantaranya meringkas, tanya-jawab, dan NLI. Semuanya mencakup lebih dari satu domain, dari sains sampai bisnis. Model base yang digunakan pada penelitian ini adalah BART (Lewis dkk., 2019) dengan input token sebesar 1024, 256 dan 512 dan LED (*Longformer Encoder-Decoder*) dengan input token 1024 dan 4069. Model BART memiliki parameter sekitar 140 juta dan LED

memiliki parameter dari 330 juta samapi 1 triliun. Dengan menggunakan metrik EM (*Exact Match*) pada dataset CNLI, model BART menghasilkan skor paling besar 77,4 dan model LED menghasilkan skor paling besar 73,4.

Ivgy dkk., (2022) menggunakan pendekatan yang dinamakan SLED (*Sliding Encoder and Decoder*) untuk memproses teks panjang. Input dipartisi menjadi potongan-potongan yang bertumpuk, lalu di-encode dengan encoder dari short-text LM (*Language Model*) dan hasilnya digabungkan kembali dengan decoder untuk menghasilkan informasi yang memiliki arti. Model dasar yang digunakan dalam tugas NLI adalah BART (Lewis dkk., 2019), T5 (Raffel dkk., 2019), dan LED (Beltagy dkk., 2020). Masing-masing model dasar diaplikasikan pendekatan SLED. Pada proses eksperimen, panjang token yang digunakan sebesar 16.000 dengan potongan berukuran 256. Optimizer yang digunakan adalah Adam, dengan learning rate dipilih dari {2e-5, 5e-5, 1e-4} dan batch size dipilih dari {8, 16, 32}. Hasilnya pada dataset CNLI dengan menggunakan metrik EM (*Exact Match*), skor paling tinggi yang dihasilkan model BART+SLED adalah 87,3

Ainslie dkk., (2023) mengenalkan CoLT5 (*Conditional Long T5*), model berbasis Transformer yang dirancang untuk menangani input panjang dengan memanfaatkan conditional computing untuk memberikan daya komputasi lebih kepada token yang lebih penting dari yang lain pada proses feedforward dan attention. Conditional computation pada CoLT5 terdiri dari 3 (tiga) komponen, yaitu routing modules, conditional feedforward layers, dan conditional attention layers. Pada proses fine-tuning, learning rate yang digunakan adalah 0,001 dengan batch size sebesar 128, dan drop out rate sebesar 0,1. Panjang input token untuk dataset CNLI adalah 8192. Besar parameter dari CoLT5 berkisar antara 433 juta samapi 5 triliun. Model CoLT5 versi large menghasilkan skor EM (*Exact Match*) sebesar 88,7 pada dataset CNLI.

Bertsch dkk., (2023) mengembangkan metode yang bernama Unlimiformer. Metode ini memanfaatkan kNN (*k-nearest-neighbor*) untuk melakukan proses decode pada tiap cross attention layer. Dengan metode ini, model dapat melakukan query pada attention layer secara sub-linear, dan secara praktis dapat menangani sekuen input yang panjangnya tak terbatas. Pada

papernya, Unlimiformer menggunakan BART-base (kurang lebih 140 parameter) sebagai base model. Hasilnya, model Unlimiformer BART-base pada tugas NLI dengan data CNLI adalah 77,7 pada skor EM (*Exact Match*). Model tidak menambah parameter baru pada base model BART-base, namun sedikit menambah waktu proses training untuk proses indexing dan searching pada kNN. Namun, saat sekuen input yang dibutuhkan lebih dari 1024 token, total waktu GPU yang dibutuhkan untuk menjalankan model hanya meningkat secara sub-linear dengan panjang input.

Tay dkk., (2023) mengenalkan paradigma pretrained model baru dengan nama UL2 (*Unified Language Learning*). UL2 memanfaatkan MoD (*Mixture of Denoisers*) untuk memungkinkan model dalam menyelesaikan berbagai jenis permasalahan pada proses pretraining dan mengimplementasikan Mode Switching untuk berganti paradigma dengan tujuan untuk lebih menyesuaikan model dengan tugas yang dilatih. UL2 memiliki total parameter kurang lebih 20 miliar dan menghasilkan skor yang menjanjikan pada tugas seperti text generation, language understanding, text classification, question answering, commonsense reasoning, long text reasoning, dan information retrieval. Hasil supervised finetuning UL2 pada data CNLI menunjukkan skor EM sebesar 88,7. Pada saat penelitian ini ditulis, skor ini merupakan SOTA pada data CNLI.

BAB 5

PENUTUP

5.1 Kesimpulan

Setelah melakukan evaluasi dari hasil eksperimen inferensi keterikatan tekstual dengan data surat kontrak perjanjian kerahasiaan CNLI (*Contract Natural Language Inference*) menggunakan model ALBERT-base dan metode *pretraining* dan *finetuning*, dapat disimpulkan hasilnya sebagai berikut.

1. Implementasi model ALBERT-base untuk melakukan inferensi keterikatan tekstual dengan data CNLI menghasilkan akurasi optimal dengan menggunakan *hyperparameter* berikut. *Batch Size* 8, *Hidden Layer Groups* 1, *Learning Rate* $2e-5$, *Epoch* 10, menggunakan *Early Stop*, dan tanpa *Warmup Steps*. Dengan mengorbankan sedikit waktu *training*, model dapat memiliki akurasi lebih tinggi menggunakan *batch size* berukuran kecil. Proses *training* dilakukan dengan 2x T4 GPU yang disediakan oleh *Kaggle.com*. *Tool* utama yang dipakai adalah PyTorch, Pandas, dan Numpy, dan Transformers oleh *Huggingface.co*.
2. Hasil dari eksperimen dievaluasi menggunakan 2 (dua) metrik, yaitu EM (*Exact Match*) dan Acc (*Accuracy*) atau akurasi. Setelah melewati beberapa proses *pretraining*, model yang diteliti menghasilkan nilai EM sebesar 85,04% dan nilai akurasi sebesar 85,0% pada subset data *testing*, lalu nilai akurasi sebesar 84,6% pada subset data *validation*. Hasil ini sudah memenuhi bahkan sedikit melebihi target akurasi yang diharapkan. Nilai akurasi dan EM dari model yang diteliti berhasil melebihi model yang didasarkan pada BERT-base dengan kurang lebih 110 juta parameter dan BART-base dengan kurang lebih 140 juta parameter, seperti *SpanNLI BERT-base*, *Unlimiformer BART-base*, dan *SCROLLS BART-base*. Namun masih kesulitan dalam menandingi model yang didasarkan pada BERT-large, BART-large, dan T5-large yang memiliki parameter kurang lebih 600 juta, seperti *SpanNLI BERT-large*, *UL20B*, *CoLT5-large*, dan *SLED BART-large*. Hasil tersebut didapat dengan menggunakan data CNLI yang telah melewati

preprocessing untuk membersihkan data. Proses *pretraining* dilakukan dengan menggunakan 2x T4 GPU dari *kaggle.com*.

5.2 Saran

Penelitian yang telah dilakukan masih memiliki banyak batasan. Karenanya, diharapkan untuk penelitian kedepannya dapat mengimplementasikan beberapa hal seperti berikut ke dalam eksperimen inferensi keterikatan tekstual dengan data surat kontrak.

1. Mengimplementasikan model ALBERT-base untuk melakukan tugas inferensi keterikatan tekstual pada tipe surat kontrak selain NDA (*Non-Disclosure Agreement* atau Surat Perjanjian Kerahasiaan), seperti surat kontrak sewa, surat kontrak kepegawaian, atau surat kontrak transaksi.
2. Meningkatkan efisiensi atau performa dari model ALBERT-base untuk mempersingkat durasi *training*, memperkecil ukuran model, atau meningkatkan kemampuan model dalam memahami konteks.
3. Melakukan *pretraining* dengan data sekunder seperti kamus atau kumpulan kata (*text corpus*) dalam ranah hukum untuk melatih model dalam memahami ciri khas dan struktur kalimat dalam ranah hukum dan surat kontrak.

DAFTAR PUSTAKA

- Ainslie, J., Lei, T., de Jong, M., Ontañón, S., Brahma, S., Zemlyanskiy, Y., Uthus, D., Guo, M., Lee-Thorp, J., Tay, Y., Sung, Y.-H., & Sanghai, S. (2023). CoLT5: Faster Long-Range Transformers with Conditional Computation. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 5085–5100. <https://doi.org/10.48550/arXiv.2303.09752>
- Allwright, S. (2022). *Micro vs Macro F1 score, what's the difference?* <https://stephenallwright.com/micro-vs-macro-f1-score/>
- Balagansky, N., & Gavrilo, D. (2022). PALBERT: Teaching ALBERT to Ponder. *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*. <http://arxiv.org/abs/2204.03276>
- Bertsch, A., Alon, U., Neubig, G., & Gormley, M. R. (2023). Unlimiformer: Long-Range Transformers with Unlimited Length Input. *NeurIPS 2023*. <https://doi.org/10.48550/arXiv.2305.01625>
- Bowman, S. R., Angeli, G., Potts, C., Manning, C. D., & Linguistics, S. (2015). A large annotated corpus for learning natural language inference. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 632–642. <https://doi.org/10.18653/v1/D15-1075>
- Cañete, J., Donoso, S., Bravo-Marquez, F., Carvallo, A., & Araujo, V. (2022). ALBETO and DistilBETO: Lightweight Spanish Language Models. Dalam N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, J. Odijk, & S. Piperidis (Ed.), *Proceedings of the Thirteenth Language Resources and Evaluation Conference* (hlm. 4291–4298). European Language Resources Association. <https://aclanthology.org/2022.lrec-1.457>
- Chalkidis, I., Androutsopoulos, I., & Michos, A. (2017). Extracting contract elements. *Proceedings of the International Conference on Artificial Intelligence and Law*, 19–28. <https://doi.org/10.1145/3086512.3086515>
- Chen, J. (2022). *Electronic Data Gathering Analysis and Retrieval: Overview, FAQ*. Investopedia. <https://www.investopedia.com/terms/e/edgar.asp#:~:text=EDGAR—Electronic%20Data%20Gathering%2C%20Analysis,required%20documents%20to%20the%20SEC.>

- Chui, M., Hall, B., Mayhew, H., Singla, A., Sukharevsky, A., & by McKinsey, A. I. (2022). The state of AI in 2022-and a half decade in review. <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai-in-2022-and-a-half-decade-in-review#/>.
- Dagan, I., Glickman, O., & Magnini, B. (2006). The PASCAL Recognising Textual Entailment Challenge. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3944 LNAI, 177–190. https://doi.org/10.1007/11736790_9
- Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- Exigent Group Limited. (2019). *Thought Leadership Contract Management Why is contract management important?* <https://cdn2.hubspot.net/hubfs/4220630/How%20GCs%20Can%20Thrive%20Not%20Just%20Survive%202019.pdf>
- Fu, Z., Avaliani, A., & Donato, M. (2023). *Energy-efficient Task Adaptation for NLP Edge Inference Leveraging Heterogeneous Memory Architectures*. <http://arxiv.org/abs/2303.16100>
- Glockner, M., Shwartz, V., & Goldberg, Y. (2018). Breaking NLI Systems with Sentences that Require Simple Lexical Inferences. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2, 650–655. <https://doi.org/10.18653/v1/P18-2103>
- Goldberg, Y. (2016). A Primer on Neural Network Models for Natural Language Processing. *Journal of Artificial Intelligence Research*, 57, 345–420. <https://doi.org/10.1613/jair.4992>
- GW University. (2023). *Common Types of Contract Documents*. <https://procurement.gwu.edu/ix-common-types-contract-documents>
- He, P., Liu, X., Gao, J., & Chen, W. (2020). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *International Conference on Learning Representations*. <http://arxiv.org/abs/2006.03654>
- Hendrycks, D., Burns, C., Chen, A., & Ball, S. (2021). CUAD: An Expert-Annotated NLP Dataset for Legal Contract Review. *35th Conference on*

Neural Information Processing Systems (NeurIPS).
<http://arxiv.org/abs/2103.06268>

- Ivgi, M., Shaham, U., & Berant, J. (2023). Efficient Long-Text Understanding with Short-Text Models. *Transactions of the Association for Computational Linguistics*, 11, 284–299. https://doi.org/10.1162/tacl_a_00547
- Just, J. (2024). Natural language processing for innovation search – Reviewing an emerging non-human innovation intermediary. *Technovation*, 129. <https://doi.org/10.1016/j.technovation.2023.102883>
- Kavlakoglu, E. (2020). *NLP vs. NLU vs. NLG: the differences between three natural language processing concepts*. IBM; Nov. <https://www.ibm.com/blog/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/>
- Khan, W., Daud, A., Khan, K., Muhammad, S., & Haq, R. (2023). Exploring the frontiers of deep learning and natural language processing: A comprehensive overview of key challenges and emerging trends. *Natural Language Processing Journal*, 4, 100026. <https://doi.org/10.1016/j.nlp.2023.100026>
- Khot, T., Sabharwal, A., & Clark, P. (2018). SCITAIL: A Textual Entailment Dataset from Science Question Answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32. <https://doi.org/10.1609/aaai.v32i1.12022>
- Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: state of the art, current trends and challenges. *Multimedia Tools and Applications*, 82(3), 3713–3744. <https://doi.org/10.1007/s11042-022-13428-4>
- Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*. <http://arxiv.org/abs/1412.6980>
- Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., & Grefenstette, E. (2017). The NarrativeQA Reading Comprehension Challenge. *Transactions of the Association for Computational Linguistics*, 6, 317–328. https://doi.org/10.1162/tacl_a_00023
- Koreeda, Y., & Manning, C. (2021). ContractNLI: A Dataset for Document-level Natural Language Inference for Contracts. Dalam M.-F. Moens, X. Huang, L. Specia, & S. W. Yih (Ed.), *Findings of the Association for Computational Linguistics: EMNLP 2021* (hlm. 1907–1919). Association

for Computational Linguistics.
<https://doi.org/10.18653/v1/2021.findings-emnlp.164>

- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *International Conference on Learning Representations*.
<https://doi.org/https://doi.org/10.48550/arXiv.1909.11942>
- Leivaditi, S., Rossi, J., & Kanoulas, E. (2020). *A Benchmark for Lease Contract Review*. <https://doi.org/10.48550/arXiv.2010.10386>
- Levesque, H. J., Davis, E., & Morgenstern, L. (2012). The Winograd Schema Challenge. *KR'12: Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, 552–561. www.aaai.org
- Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.
<https://doi.org/10.18653/v1/2020.acl-main.703>
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *International Conference on Learning Representations*. <http://arxiv.org/abs/1907.11692>
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems*, 26. <https://doi.org/10.48550/arXiv.1310.4546>
- Naseem, U., Khushi, M., Reddy, V., Rajendran, S., Razzak, I., & Kim, J. (2021). BioALBERT: A Simple and Effective Pre-trained Language Model for Biomedical Named Entity Recognition. *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–7.
<https://doi.org/10.1109/IJCNN52387.2021.9533884>
- Openai, A. R., Openai, K. N., Openai, T. S., & Openai, I. S. (2018). *Improving Language Understanding by Generative Pre-Training*. <https://gluebenchmark.com/leaderboard>
- Ostermann, S., Roth, M., & Pinkal, M. (2019). MCScript2.0: A Machine Comprehension Corpus Focused on Script Events and Participants. *Proceedings of the Eighth Joint Conference on Lexical and*

- Computational Semantics (*SEM)*, 103–117.
<https://doi.org/10.18653/v1/S19-1012>
- Radack, D. V. (1994). *Understanding Confidentiality Agreements*. JOM. The Minerals, Metals & Materials Society.
<https://www.tms.org/pubs/journals/JOM/matters/matters-9405.html>
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 1–67. <http://arxiv.org/abs/1910.10683>
- Rashkin, H., Sap, M., Allaway, E., Smith, N. A., & Choi, Y. (2018). Event2Mind: Commonsense Inference on Events, Intents, and Reactions. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 1, 463–473. <https://doi.org/10.18653/v1/P18-1043>
- Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. <https://doi.org/10.48550/arXiv.1910.01108>
- Schmitt, M., & Schütze, H. (2019). SherLIiC: A Typed Event-Focused Lexical Inference Benchmark for Evaluating Natural Language Inference. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 902–914. <https://doi.org/10.18653/v1/P19-1086>
- Servan, C., Ghannay, S., & Rosset, S. (2024). mALBERT: Is a Compact Multilingual BERT Model Still Worth It? Dalam N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, & N. Xue (Ed.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)* (hlm. 11023–11029). ELRA and ICCL. <https://aclanthology.org/2024.lrec-main.960>
- Shaham, U., Segal, E., Ivgi, M., Efrat, A., Yoran, O., Haviv, A., Gupta, A., Xiong, W., Geva, M., Berant, J., & Levy, O. (2022). SCROLLS: Standardized CompaRison Over Long Language Sequences. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 12007–12021. <http://arxiv.org/abs/2201.03533>
- Simplilearn. (2023). *What is PyTorch, and How Does It Work: All You Need to Know*. <https://builtin.com/machine-learning/pytorch#:~:text=PyTorch%20is%20an%20open-source,for%20training%20deep%20neural%20networks.&text=PyTorch>

%20provides%20a%20way%20to,to%20grow%20in%20popularity%20C%20too.

- Storks, S., Gao, Q., & Chai, J. Y. (2020). *Recent Advances in Natural Language Inference: A Survey of Benchmarks, Resources, and Approaches*. <https://doi.org/10.48550/arXiv.1904.01172>
- Talmor, A., Yoran, O., Bras, R. Le, Bhagavatula, C., Goldberg, Y., Choi, Y., & Berant, J. (2022). CommonsenseQA 2.0: Exposing the Limits of AI through Gamification. *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*. <http://arxiv.org/abs/2201.05320>
- Tay, Y., Dehghani, M., Tran, V. Q., Garcia, X., Wei, J., Wang, X., Chung, H. W., Shakeri, S., Bahri, D., Schuster, T., Zheng, H. S., Zhou, D., Houlsby, N., & Metzler, D. (2022). UL2: Unifying Language Learning Paradigms. *ICLR 2023 Conference*. <http://arxiv.org/abs/2205.05131>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *31st Conference on Neural Information Processing Systems (NIPS)*. <http://arxiv.org/abs/1706.03762>
- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, 353–355. <https://doi.org/10.18653/v1/W18-5446>
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., ... Rush, A. (2020). Transformers: State-of-the-Art Natural Language Processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. <https://doi.org/10.18653/v1/2020.emnlp-demos.6>
- Xiao, C., Zhong, H., Guo, Z., Tu, C., Liu, Z., Sun, M., Feng, Y., Han, X., Hu, Z., Wang, H., & Xu, J. (2018). *CAIL2018: A Large-Scale Legal Dataset for Judgment Prediction*. <https://doi.org/10.48550/arXiv.1807.02478>
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *NIPS'19: Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 5753–5763. <http://arxiv.org/abs/1906.08237>

- Zellers, R., Bisk, Y., Schwartz, R., & Choi, Y. (2018). SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 93–104. <http://arxiv.org/abs/1808.05326>
- Zhang, L., Wilson, S. R., & Mihalcea, R. (2018). Multi-Label Transfer Learning for Multi-Relational Semantic Similarity. *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM)*, 44–50. <https://doi.org/10.18653/v1/S19-1005>

LAMPIRAN

Lampiran 1. Biodata Penulis



Abdillah Azmi merupakan penulis dari skripsi ini. Lahir di Wonosobo pada 17 Mei 2002, dari Ibu Anik Rofiana dan Bapak Ismanto. Anak pertama dari dua bersaudara. Besar di pelosok Kalimantan Tengah dan memutuskan untuk menuntut ilmu ke pulau Jawa. Menempuh pendidikan formal di SMP Kridatama Lancar 1 dan SMK Negeri 1 Wonosobo dengan fokus di *Software Engineering*. Saat penelitian ini dilakukan, penulis berdomisili di Sekaran, Kec. Gunungpati, Kota Semarang, Provinsi Jawa Tengah, dan tengah menempuh pendidikan Sarjana (S1) di Program Studi Teknik Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Semarang.

Lampiran 2. SK Pembimbing



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
UNIVERSITAS NEGERI SEMARANG
FAKULTAS MATEMATIKA DAN ILMU
PENGETAHUAN ALAM

Gedung D12, Kampus Sekaran, Gunungpati,
 Semarang 50229.
 Telepon: 024 - 8508112
 Laman: mipa.unnes.ac.id, surel:
 mipa@mail.unnes.ac.id

KEPUTUSAN
DEKAN FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI SEMARANG
Nomor: B/382/UN37.1.4/TD.06/2024
Tentang
PENETAPAN DOSEN PEMBIMBING SKRIPSI/TUGAS AKHIR SEMESTER GASAL/GENAP
TAHUN AKADEMIK 2023/2024

- Menimbang : Bahwa untuk memperlancar mahasiswa Prodi Teknik Informatika, S1 Fakultas Matematika dan Ilmu Pengetahuan Alam membuat Skripsi/Tugas Akhir, maka perlu menetapkan Dosen-dosen Jurusan/Prodi Ilmu Komputer/Teknik Informatika, S1 Fakultas Matematika dan Ilmu Pengetahuan Alam UNNES untuk menjadi pembimbing.
- Mengingat : 1. Undang-undang No.20 Tahun 2003 tentang Sistem Pendidikan Nasional (Tambahan Lembaran Negara RI No.4301, penjelasan atas Lembaran Negara RI Tahun 2003, Nomor 78)
 2. Peraturan Rektor No. 21 Tahun 2011 tentang Sistem Informasi Skripsi UNNES
 3. SK. Rektor UNNES No. 164/O/2004 tentang Pedoman penyusunan Skripsi/Tugas Akhir Mahasiswa Strata Satu (S1) UNNES;
 4. SK Rektor UNNES No.162/O/2004 tentang penyelenggaraan Pendidikan UNNES;
- Menimbang : Usulan Koordinator Prodi Teknik Informatika, S1 Tanggal 22 Februari 2024
- MEMUTUSKAN**
- Menetapkan :
 PERTAMA : Menunjuk dan menugaskan kepada:
 Nama : Dr. Alamsyah, S.Si., M.Kom.
 NIP : 197405172006041001
 Pangkat/Golongan : Pembina - IV/a
 Jabatan Akademik : Lektor Kepala
 Sebagai Pembimbing
 Untuk membimbing mahasiswa penyusun skripsi/Tugas Akhir :
 Nama : Abdillah Azmi
 NIM : 4611420020
 Prodi : Teknik Informatika, S1
 Topik : Natural Language Processing, Natural Language Inference
- KEDUA : Keputusan ini mulai berlaku sejak tanggal ditetapkan.

Tembusan
 1. WD Bid. Akademik dan Mawa
 2. Ketua Prodi
 3. Peringgal



4611420020

...: FM-03-AKD-24/Rev. 00 ...:

Dikirim secara elektronik menggunakan sertifikat elektronik yang diterbitkan oleh Balai Sertifikasi Elektronik (BSrE), BSSN.

DITETAPKAN DI : SEMARANG
 PADA TANGGAL : 22 Februari 2024
 DEKAN



Prof. Dr. Edy Cahyono, M.Si.
 NIP 196412051990021001

Lampiran 3. SK Penguji



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
UNIVERSITAS NEGERI SEMARANG
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN
ALAM

Gedung D5, Kampus Sekaran, Gunungpati,
 Semarang 50229.
 Telepon: 0248508328
 Laman: <http://ilkom.unnes.ac.id>, surel:
 ilkom@mail.unnes.ac.id

No. : B/7897/UN37.1.4/PK.03.00/2024
 Lamp. :
 Hal : Surat Tugas Panitia Ujian Sarjana

Dengan ini kami tetapkan bahwa ujian Sarjana Fakultas Matematika dan Ilmu Pengetahuan Alam UNNES untuk program studi Teknik Informatika, S1 adalah sebagai berikut:

I. Susunan Panitia Ujian:

a. Ketua	: Prof. Dr. Edy Cahyono, M.Si.
b. Sekretaris	: Dr. Alamsyah, S.Si., M.Kom.
c. Pembimbing Utama	: Dr. Alamsyah, S.Si., M.Kom.
d. Penguji	: 1. Endang Sugiharti, S.Si., M.Kom. : 2. Yahya Nur Ifriza, S.Pd., M.Kom.

II. Calon yang diuji:

Nama	: Abdillah Azmi
NIM/Program Studi	: 4611420020 /Teknik Informatika, S1
Judul Skripsi	: PENENTUAN KETERIKATAN TEKSTUAL DALAM SURAT KONTRAK BERTIPE NDA (NON-DISCLOSURE AGREEMENT) MENGGUNAKAN ALBERT (A LITE BERT)

II. Waktu dan Tempat Ujian:

Hari/Tanggal	: Kamis, 16 Mei 2024
Jam	: 13:00:00
Tempat	: D5 Lantai 3
Pakaian	:

Semarang, 7 Mei 2024
 Dekan,



Tembusan
 1. Koordinator Program Studi Teknik Informatika, S1
 2. Calon yang diuji

Prof. Dr. Edy Cahyono, M.Si.
 NIP 196412051990021001



4611420020

Dokumen ini telah ditandatangani secara elektronik menggunakan sertifikat elektronik yang diterbitkan oleh Balai Sertifikasi Elektronik (BSrE), BSSN.

Lampiran 4. Source Code

```

1. import pandas as pd
2. import json
3. import torch
4. import random
5. from sklearn.metrics import f1_score
6. import numpy as np
7.
8. BATCH_SIZE = 8 #16 #20 #8
9. WARMUP_STEPS = 0 #100
10. NUM_HIDDEN_GROUPS = 1
11. NUM_EPOCHS = 10
12. LEARNING_RATE = 2e-5 #2e-5 #2e-6
13. MAX_TOKEN_LEN = 512
14. SEED = 444
15. EARLY_STOP = True
16. NAME_FILE =
f'BS{BATCH_SIZE}_WM{WARMUP_STEPS}_HG{NUM_HIDDEN_GROUPS}_EP{NUM_EPOCHS}_LR{LEARNING_R
ATE}_ES{EARLY_STOP}_S{SEED}'
17.
18. def set_seed(seed):
19.     random.seed(seed)
20.     np.random.seed(seed)
21.     torch.manual_seed(seed)
22.     torch.cuda.manual_seed_all(seed)
23.     torch.backends.cudnn.deterministic = True
24.     torch.backends.cudnn.benchmark = False
25.
26. set_seed(SEED)
27.
28. device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
29. data = pd.read_json("https://raw.githubusercontent.com/ab-
azmi/skripsi/main/data/contract_nli_v1.jsonl", lines=True)
30.
31. train_df = data[data['subset'] == 'train'] # 70%
32. val_df = data[data['subset'] == 'dev'] # 10%
33. test_df = data[data['subset'] == 'test'] # 20%
34.
35. from torch.utils.data import Dataset, TensorDataset, DataLoader
36. from torch.nn.utils.rnn import pad_sequence
37. import pickle
38. import os
39. from transformers import AlbertTokenizer
40.
41. class CNLIDataAlbert(Dataset):
42.     def __init__(self, train_df, val_df, test_df, max_len):
43.         self.label_dict = {'entailment':0, 'contradiction':1, 'neutral':2}
44.         self.train_df = train_df
45.         self.val_df = val_df
46.         self.test_df = test_df
47.         self.max_len = max_len
48.
49.         self.tokenizer = AlbertTokenizer.from_pretrained('albert-base-v2',
do_lower_case=True)
50.         self.train_data = None
51.         self.val_data = None
52.         self.test_data = None
53.         self.init_data()
54.
55.     def init_data(self):
56.         self.train_data = self.load_data(self.train_df)
57.         self.val_data = self.load_data(self.val_df)
58.         self.test_data = self.load_data(self.test_df)
59.
60.     def load_data(self, df):
61.         max_len = self.max_len

```

```

62.         token_ids = []
63.         mask_ids = []
64.         seg_ids = []
65.         y = []
66.
67.         premise_list = df['premise'].to_list()
68.         hypo_list = df['hypothesis'].to_list()
69.         label_list = df['label'].to_list()
70.
71.         for (premise, hypothesis, label) in zip(premise_list, hypo_list,
label_list):
72.             #Encoding
73.             premise_id = self.tokenizer.encode(premise,
add_special_tokens=False, max_length=max_len, truncation=True)
74.             hypo_id = self.tokenizer.encode(hypothesis,
add_special_tokens=False, max_length=max_len, truncation=True)
75.
76.             pair_token_ids = [self.tokenizer.cls_token_id] + premise_id +
[self.tokenizer.sep_token_id] + hypo_id + [self.tokenizer.sep_token_id]
77.             premise_len = len(premise_id)
78.             hypo_len = len(hypo_id)
79.
80.             #check if token length exceed limit
81.             if len(pair_token_ids) > max_len:
82.                 # Split the tokens into chunks
83.                 chunks = [pair_token_ids[i:i+max_len] for i in range(0,
len(pair_token_ids), max_len)]
84.
85.                 for chunk in chunks:
86.                     segment_ids = torch.tensor([0] * (len(chunk)//2) + [1] *
((len(chunk))//2)) #premise & hypos
87.                     attention_mask_ids = torch.tensor([1]* len(chunk)) #mask
padded values
88.
89.                     token_ids.append(torch.tensor(chunk))
90.                     seg_ids.append(segment_ids)
91.                     mask_ids.append(attention_mask_ids)
92.                     y.append(self.label_dict[label])
93.                 else:
94.                     segment_ids = torch.tensor([0] * (premise_len+2) + [1] *
(hypo_len+1)) #premise & hypos
95.                     attention_mask_ids = torch.tensor([1]* (premise_len + hypo_len
+ 3)) #mask padded values
96.
97.                     token_ids.append(torch.tensor(pair_token_ids))
98.                     seg_ids.append(segment_ids)
99.                     mask_ids.append(attention_mask_ids)
100.                    y.append(self.label_dict[label])
101.
102.            token_ids = pad_sequence(token_ids, batch_first=True)
103.            mask_ids = pad_sequence(mask_ids, batch_first=True)
104.            seg_ids = pad_sequence(seg_ids, batch_first=True)
105.            y = torch.tensor(y)
106.            dataset = TensorDataset(token_ids, mask_ids, seg_ids, y)
107.            print(len(dataset))
108.            return dataset
109.
110.        def get_data_loaders(self, batch_size=8, shuffle=True):
111.            train_loader = DataLoader(
112.                self.train_data,
113.                shuffle=shuffle,
114.                batch_size=batch_size
115.            )
116.            val_loader = DataLoader(
117.                self.val_data,
118.                shuffle=shuffle,
119.                batch_size=batch_size

```



```

120.         )
121.         test_loader = DataLoader(
122.             self.test_data,
123.             shuffle=False,
124.             batch_size=batch_size
125.         )
126.         return train_loader, val_loader, test_loader
127.
128.
129. cnli_dataset = CNLIDataAlbert(train_df, val_df, test_df, max_len=MAX_TOKEN_LEN)
130. train_loader, val_loader, test_loader =
cnli_dataset.get_data_loaders(batch_size=BATCH_SIZE)
131.
132. from transformers import AlbertForSequenceClassification, AlbertConfig
133.
134. if NUM_HIDDEN_GROUPS > 1:
135.     # Initialize the configuration
136.     config = AlbertConfig(
137.         num_hidden_layers=12, # Total number of layers
138.         num_attention_heads=12,
139.         hidden_size=768,
140.         intermediate_size=3072,
141.         num_labels=3,
142.         num_hidden_groups=NUM_HIDDEN_GROUPS, # Number of layer groups
143.     )
144.     model = AlbertForSequenceClassification.from_pretrained("albert-base-v2",
config=config)
145. else:
146.     model = AlbertForSequenceClassification.from_pretrained("albert-base-v2",
num_labels=3)
147.
148. model.to(device)
149.
150. TOTAL_TRAIN_STEP = len(train_loader)
151.
152. #Warmup step
153. def warmup(current_step: int):
154.     if current_step < WARMUP_STEPS:
155.         return float((current_step + 1) / WARMUP_STEPS)
156.     else:
157.         return max(0.0, float(TOTAL_TRAIN_STEP - current_step) / float(max(1,
TOTAL_TRAIN_STEP - WARMUP_STEPS)))
158.
159. param_optimizer = list(model.named_parameters())
160. no_decay = ['bias', 'gamma', 'beta']
161. optimizer_grouped_parameters = [
162.     {'params': [p for n, p in param_optimizer if not any(nd in n for nd in
no_decay)], 'weight_decay': 0.01},
163.     {'params': [p for n, p in param_optimizer if any(nd in n for nd in
no_decay)], 'weight_decay': 0.0},
164. ]
165.
166. from torch.optim import AdamW
167.
168. optimizer = AdamW(optimizer_grouped_parameters, lr=LEARNING_RATE)
169. lr_scheduler = torch.optim.lr_scheduler.LambdaLR(optimizer, lr_lambda=warmup)
170.
171. def multi_acc(y_pred, y_test):
172.     acc = (torch.log_softmax(y_pred, dim=1).argmax(dim=1) ==
y_test).sum().float() / float(y_test.size(0))
173.     return acc
174.
175. import time
176.
177. def train(model, train_loader, val_loader, optimizer):
178.     total_step = len(train_loader)
179.

```

```

180. #Train
181. best_loss = float('inf')
182. patience = 2
183. patience_counter = 0
184.
185.     for epoch in range(NUM_EPOCHS):
186.         start = time.time()
187.         model.train()
188.         total_train_loss = 0
189.         total_train_acc = 0
190.         for batch_idx, (pair_token_ids, mask_ids, seg_ids, y) in
enumerate(train_loader):
191.             optimizer.zero_grad()
192.             pair_token_ids = pair_token_ids.to(device)
193.             mask_ids = mask_ids.to(device)
194.             seg_ids = seg_ids.to(device)
195.             labels = y.to(device)
196.
197.             loss, prediction = model(pair_token_ids, token_type_ids=seg_ids,
attention_mask=mask_ids, labels=labels).values()
198.
199.             acc = multi_acc(prediction, labels)
200.
201.             loss.backward()
202.             optimizer.step()
203.             lr_scheduler.step()
204.
205.             total_train_loss += loss.item()
206.             total_train_acc += acc.item()
207.
208.         train_acc = total_train_acc/len(train_loader)
209.         train_loss = total_train_loss/len(train_loader)
210.
211.         model.eval()
212.         total_val_acc = 0
213.         total_val_loss = 0
214.         with torch.no_grad():
215.             for batch_idx, (pair_token_ids, mask_ids, seg_ids, y) in
enumerate(val_loader):
216.                 optimizer.zero_grad()
217.                 pair_token_ids = pair_token_ids.to(device)
218.                 mask_ids = mask_ids.to(device)
219.                 seg_ids = seg_ids.to(device)
220.                 labels = y.to(device)
221.
222.                 loss, prediction = model(pair_token_ids,
token_type_ids=seg_ids, attention_mask=mask_ids, labels=labels).values()
223.
224.                 acc = multi_acc(prediction, labels)
225.
226.                 total_val_loss += loss.item()
227.                 total_val_acc += acc.item()
228.
229.             val_acc = total_val_acc/len(val_loader)
230.             val_loss = total_val_loss/len(val_loader)
231.
232.             end = time.time()
233.             hours, rem = divmod(end-start, 3600)
234.             minutes, seconds = divmod(rem, 60)
235.
236.             print(f'Epoch {epoch+1}: train_loss: {train_loss:.4f} train_acc:
{train_acc:.4f} | val_loss: {val_loss:.4f} val_acc: {val_acc:.4f}')
237.
238. print("{:0>2}:{:0>2}:{:05.2f}".format(int(hours),int(minutes),seconds))
239.
240.     #Early stops
241.     if val_loss < best_loss:

```

```

241.         best_loss = val_loss
242.         torch.save(model.state_dict(), f'{NAME_FILE}.pth')
243.         patience_counter = 0
244.     else:
245.         patience_counter += 1
246.
247.         # If no improvement is observed for `patience` epochs, stop training
248.         if patience_counter >= patience:
249.             break
250.
251.
252. train(model, train_loader, val_loader, optimizer)
253.
254. #Test
255. def test(model, test_loader):
256.     model.eval()
257.     total_test_loss = 0
258.     total_test_acc = 0
259.     true_labels = []
260.     predicted_labels = []
261.     with torch.no_grad():
262.         for batch_idx, (pair_token_ids, mask_ids, seg_ids, y) in
enumerate(test_loader):
263.             pair_token_ids = pair_token_ids.to(device)
264.             mask_ids = mask_ids.to(device)
265.             seg_ids = seg_ids.to(device)
266.             labels = y.to(device)
267.
268.             loss, prediction = model(pair_token_ids, token_type_ids=seg_ids,
attention_mask=mask_ids, labels=labels).values()
269.
270.             predicted_labels.extend(prediction.cpu().numpy().tolist())
271.             true_labels.extend(y.cpu().numpy().tolist())
272.
273.             acc = multi_acc(prediction, labels)
274.
275.             total_test_loss += loss.item()
276.             total_test_acc += acc.item()
277.
278.     test_acc = total_test_acc/len(test_loader)
279.     test_loss = total_test_loss/len(test_loader)
280.
281.     #turn predicted labels from 1 hot to 1d array
282.     predicted_labels = np.argmax(predicted_labels, axis=1)
283.
284.     #f1 for entail [0]
285.     f1_entail = f1_score(true_labels, predicted_labels, average='micro',
labels=[0])
286.
287.     #f1 for contradiction [1]
288.     f1_contradiction = f1_score(true_labels, predicted_labels,
average='micro', labels=[1])
289.
290.     #f1 for neutral [2]
291.     f1_neutral = f1_score(true_labels, predicted_labels, average='micro',
labels=[2])
292.
293.     #f1 for all
294.     f1_all = f1_score(true_labels, predicted_labels, average='micro')
295.
296.     #f1 macro
297.     f1_macro = f1_score(true_labels, predicted_labels, average='macro')
298.
299.
300.     print(f'Test: test_loss: {test_loss:.4f} test_acc: {test_acc:.4f}')
301.     print(f'Test: F1 Micro Entail: {f1_entail:.4f}')
302.     print(f'Test: F1 Micro Contradiction: {f1_contradiction:.4f}')

```

```
303.         print(f'Test: F1 Micro Neutral: {f1_neutral:.4f}')
304.         print(f'Test: F1 Micro All: {f1_all:.4f}')
305.         print(f'Test: F1 Macro: {f1_macro:.4f}')
306.
307.     return true_labels, predicted_labels
308.
309. true, pred = test(model, test_loader)
310.
```