

**IDENTIFIKASI EKSPRESI IDIOMATIK DAN VALIDASINYA
MENGUNAKAN *TRUTH DISCOVERY***

SKRIPSI



NI MADE YULI CAHYANI

NIM. 1808561027

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS UDAYANA**

JIMBARAN

2022

**IDENTIFIKASI EKSPRESI IDIOMATIK DAN VALIDASINYA
MENGUNAKAN *TRUTH DISCOVERY***

SKRIPSI



NI MADE YULI CAHYANI

NIM. 1808561027

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS UDAYANA**

JIMBARAN

2022

SURAT PERNYATAAN KEASLIAN KARYA ILMIAH

Yang bertanda tangan di bawah ini menyatakan bahwa naskah Skripsi dengan judul:
**“IDENTIFIKASI EKSPRESI IDIOMATIK DAN VALIDASINYA
MENGUNAKAN *TRUTH DISCOVERY*”.**

Nama : Ni Made Yuli Cahyani
NIM : 1808561027
Program Studi : Informatika
E-mail : yulicahyani1101@gmail.com
Nomor telp/HP : 085647065215
Alamat : Jln. Pantai Pererenan No. 42 Br. Batu, Desa Pererenan, Kec.
Mengwi, Kab. Badung, Bali

Belum pernah dipublikasikan dalam dokumen skripsi, jurnal nasional maupun internasional atau dalam prosiding manapun, dan tidak sedang atau akan diajukan untuk publikasi di jurnal atau prosiding manapun. Apabila di kemudian hari terbukti terdapat pelanggaran kaidah-kaidah akademik pada karya ilmiah saya, maka saya bersedia menanggung sanksi-sanksi yang dijatuhkan karena kesalahan tersebut, sebagaimana diatur oleh Peraturan Menteri Pendidikan Nasional Nomor 17 Tahun 2010 tentang Pencegahan dan Penanggulangan Plagiat di Perguruan Tinggi.

Demikian Surat Pernyataan ini saya buat dengan sesungguhnya untuk dapat dipergunakan bilamana diperlukan.

Bukit Jimbaran, 11 April 2022

Yang membuat pernyataan,

Ni Made Yuli Cahyani

NIM. 1808561027

LEMBAR PENGESAHAN TUGAS AKHIR

Judul : Identifikasi Ekspresi Idiomatik dan Validasinya
Menggunakan *Truth Discovery*
Nama : Ni Made Yuli Cahyani
NIM : 1808561027
Tanggal Seminar : 11 April 2022

Disetujui oleh:

Pembimbing I



Dr. Anak Agung Istri Ngurah Eka
Karyawati, S.Si., M.Eng.
NIP. 197404071998022001

Penguji I



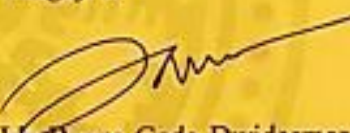
Agus Muliantara, S.Kom, M.Kom.
NIP. 198006162005011001

Pembimbing II



Luh Arida Ayu Rahning Putri, S.Kom.,
M.Cs.
NIP. 198209182008122002

Penguji II



Ida Bagus Gede Dwidasmaru, S.Kom.,
M.Cs.
NIP. 198503152010121007

Penguji III



Dra. Luh Gede Astuti, M.Kom.
NIP. 196401141994022001

Mengetahui,
Program Studi Informatika
FMIPA UNUD
KPS,



Dr. Ir. I Ketut Gede Suhartana, S.Kom., M.Kom.
NIP. 197201102008121001

Judul : Identifikasi Ekspresi Idiomatik dan Validasinya
Menggunakan *Truth Discovery*

Nama : Ni Made Yuli Cahyani (1808561027)

Pembimbing : 1. Dr. Anak Agung Istri Ngurah Eka Karyawati, S.Si.,
M.Eng.
2. Luh Arida Ayu Rahning Putri, S.Kom., M.Cs.

ABSTRAK

Ekspresi idiomatik merupakan frasa yang terdiri dari urutan dua kata atau lebih yang memiliki makna yang tidak dapat diprediksi dari makna kata-kata individu penyusunnya. Ekspresi idiomatik ada di hampir semua bahasa tetapi sulit untuk diekstrak karena tidak ada algoritma yang dapat secara tepat menguraikan struktur dari ekspresi idiomatik, sehingga sebagian besar sistem mesin terjemahan berbasis aturan umumnya menerjemahkan ekspresi idiomatik dengan cara menerjemahkan kata demi kata penyusunnya, namun hasil terjemahan tidak menghasilkan makna yang sebenarnya dari ekspresi idiomatik. Berdasarkan permasalahan tersebut, penulis mencoba melakukan penelitian mengenai identifikasi penggunaan ekspresi idiomatik pada kalimat bahasa Indonesia. Pertama-tama penulis akan melakukan proses klasifikasi kalimat menggunakan BERT untuk mengetahui apakah kalimat mengandung ekspresi idiomatik atau tidak. Selanjutnya ekspresi idiomatik diidentifikasi berdasarkan pendekatan berbasis semantik distribusi dan kemudian divalidasi secara otomatis menggunakan metode *Truth Discovery*. Dari penelitian yang dilakukan, proses klasifikasi menggunakan *Bidirectional Encoder Representations from Transformers* (BERT) memperoleh akurasi sebesar 0,98; presisi 0,99; *recall* 0,97 dan *f1-score* 0,98 dengan *learning rate* $2e-5$ dan minimum *epoch* 6. Kemudian Identifikasi ekspresi idiomatik pada kalimat berbahasa Indonesia menggunakan *Truth Discovery* memperoleh akurasi sebesar 0,82; presisi 1,0; *recall* 0,64 dan *f1-score* 0,78.

Kata kunci: Ekspresi Idiomatik, BERT, *Truth Discovery*, Validasi, Semantik Distribusi

Title : Identification of Idiomatic Expressions and Its Validation
Using Truth Discovery

Name : Ni Made Yuli Cahyani (1808561027)

Supervisor : 1. Dr. Anak Agung Istri Ngurah Eka Karyawati, S.Si.,
M.Eng.
2. Luh Arida Ayu Rahning Putri, S.Kom., M.Cs.

ABSTRACT

Idiomatic expressions are phrases that consist of a sequence of two or more words that have a meaning that cannot be predicted from the meaning of the individual words that compose it. Idiomatic expressions exist in almost all languages but are difficult to extract because there is no algorithm that can precisely decipher the structure of idiomatic expressions, so most rule-based machine translation systems generally translate idiomatic expressions by translating word for word their constituents, but the translation results do not produce the true meaning of the idiomatic expression. Based on this problem, the author tries to do research on the identification of the use of idiomatic expressions in Indonesian sentences. First, the author conducts the sentence classification process using BERT to find out whether the sentence contains idiomatic expressions or not. Furthermore, idiomatic expressions are identified based on distributional semantic based approach and then validated automatically using the Truth Discovery method. From the research conducted, the classification process using Bidirectional Encoder Representations from Transformers (BERT) obtained an accuracy of 0.97; precision 0.96; recall 0.98 and f1-score 0.97 with a learning rate of $2e-5$ and minimum epoch 6. Then the identification of idiomatic expressions in Indonesian sentences using Truth Discovery obtained an accuracy of 0.82; precision 1.0; recall 0.64 and f1-score 0.78.

Keywords: Idiomatic Expressions, BERT, Truth Discovery, Validation, Distribution Semantic

KATA PENGANTAR

Penelitian dengan judul “Identifikasi Ekspresi Idiomatik dan Validasinya Menggunakan *Truth Discovery*” ini disusun dalam rangkaian kegiatan pelaksanaan Tugas Akhir di Program Studi Informatika FMIPA UNUD. Penelitian ini dilaksanakan pada periode Januari 2022 hingga Juni 2022 di Universitas Udayana.

Sehubungan dengan telah terselesaikannya penelitian ini, maka diucapkan terima kasih dan penghargaan kepada berbagai pihak yang telah membantu penulis, antara lain.

1. Ibu Dr. Anak Agung Istri Ngurah Eka Karyawati, S.Si., M.Eng. selaku Pembimbing I yang dengan penuh dedikasi telah banyak meluangkan waktu untuk membantu pelaksanaan penelitian ini;
2. Ibu Luh Arida Ayu Rahning Putri, S.Kom., M.Cs. selaku Pembimbing II yang telah bersedia mengkritisi, memeriksa, dan menyempurnakan penulisan tugas akhir ini;
3. Bapak Cokorda Rai Adi Pramatha, ST.MM.PhD selaku Pembimbing Akademik yang telah memberikan saran-saran dan motivasi selama penulis menjalankan proses perkuliahan;
4. Bapak-bapak dan Ibu-ibu dosen Program Studi Informatika yang telah meluangkan waktu turut memberikan saran dan masukan dalam pelaksanaan penelitian;
5. Teman-teman dan keluarga yang telah membantu dan memberikan semangat serta dukungan moral dalam penyelesaian tugas akhir ini;

Dalam penelitian ini, disadari pula bahwa sudah tentu hasil-hasil dari penelitian ini masih mengandung kelemahan dan kekurangan. Memperhatikan hal ini, maka masukan dan saran-saran penyempurnaan sangat diharapkan.

Bukit Jimbaran, 11 April 2022

Penulis,

Ni Made Yuli Cahyani

DAFTAR ISI

LEMBAR JUDUL	i
SURAT PERNYATAAN KEASLIAN KARYA ILMIAH.....	ii
LEMBAR PENGESAHAN TUGAS AKHIR	iii
ABSTRAK	iv
ABSTRACT	v
KATA PENGANTAR	vi
DAFTAR ISI.....	vii
DAFTAR TABEL	x
DAFTAR GAMBAR	xii
DAFTAR LAMPIRAN	xiv
BAB I PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah.....	3
1.4 Tujuan Penelitian.....	3
1.5 Manfaat Penelitian.....	3
1.6 Sistematika Penulisan.....	4
BAB II TINJAUAN PUSTAKA.....	5
2.1 Tinjauan Teori	5
2.1.1 Kelas Kata Bahasa Indonesia	5
2.1.2 Ekspresi Idiomatik Bahasa Indonesia	6
2.1.3 <i>Text Preprocessing</i>	6
2.1.4 <i>Part-of-Speech (POS) Tagging</i>	7
2.1.5 <i>Chunking</i>	8
2.1.6 <i>Bidirectional Encoder Representations from Transformers (BERT)</i>	9
2.1.7 <i>Distributional Semantic Based Approach</i>	16
2.1.8 <i>Truth Discovery</i>	18
2.1.9 <i>Confusion Matrix</i>	19
2.1.10 Metode Pengembangan Perangkat Lunak <i>Waterfall</i>	20

2.2	Tinjauan Empiris	21
BAB III ANALISIS DAN PERANCANGAN SISTEM		25
3.1	Metode Pengembangan Sistem.....	25
3.1.1	Analisa Kebutuhan.....	25
3.1.2	Perancangan Antarmuka Sistem	26
3.1.3	Pengkodean Program	28
3.1.4	Pengujian.....	28
3.1.5	Penerapan dan Pemeliharaan.....	28
3.2	Data dan Metode Pengumpulan Data	28
3.3	Desain Sistem	30
3.3.1	<i>Text Preprocessing</i>	30
3.3.2	Klasifikasi Kalimat Menggunakan BERT	32
3.3.3	<i>POS Tagging</i>	33
3.3.4	<i>Chunking</i>	34
3.3.5	Identifikasi Ekspresi Idiomatik Menggunakan <i>Distributional Semantic Based Approach</i>	34
3.3.6	Validasi Ekspresi Idiomatik Menggunakan <i>Truth Discovery</i>	36
3.4	Desain Evaluasi Sistem	37
3.4.1	Skenario Pengujian.....	37
3.4.2	Evaluasi Sistem	39
BAB IV HASIL DAN PEMBAHASAN		40
4.1	Implementasi Sistem	40
4.1.1	Implementasi <i>Text Preprocessing</i>	40
4.1.2	Implementasi Klasifikasi Kalimat Menggunakan BERT.....	40
4.1.3	Implementasi <i>POS Tagging</i>	45
4.1.4	Implementasi <i>Chunking</i>	46
4.1.5	Implementasi Identifikasi Ekspresi Idiomatik Menggunakan <i>Distributional Semantic Based Approach</i>	47
4.1.6	Implementasi Validasi Ekspresi Idiomatik Menggunakan <i>Truth Discovery</i>	49
4.1.7	Implementasi Tampilan Antarmuka.....	53

4.2	Hasil Penelitian.....	55
4.2.1	Validasi Model BERT untuk Klasifikasi Kalimat yang Mengandung Ekspresi Idiomatik	55
4.2.2	Evaluasi Model BERT untuk Klasifikasi Kalimat yang Mengandung Ekspresi Idiomatik Menggunakan Data Baru	60
4.2.3	Evaluasi Identifikasi Ekspresi Idiomatik Menggunakan <i>Truth Discovery</i>	61
BAB V KESIMPULAN DAN SARAN.....		64
5.1	Kesimpulan.....	64
5.2	Saran	64
DAFTAR PUSTAKA		66
LAMPIRAN		69

DAFTAR TABEL

Tabel	Halaman
Tabel 2. 2 Part of Speech Tagset Bahasa Indonesia.....	5
Tabel 2. 1 Contoh Ekspresi Idiomatik Bahasa Indonesia	6
Tabel 2. 3 Confusion Matrix	19
Tabel 3. 1 Kebutuhan Fungsional Sistem	25
Tabel 3. 2 Kebutuhan Non-Fungsional Sistem	25
Tabel 3. 3 Indonesian Manually Tagged Corpus	29
Tabel 3. 4 Contoh Kalimat Bahasa Indonesia	29
Tabel 3. 5 Contoh Data Situs Web.....	29
Tabel 3. 6 Contoh Punctuation Removal	31
Tabel 3. 7 Contoh Tokenization.....	31
Tabel 3. 8 Contoh Case Conversion.....	31
Tabel 3. 9 Contoh BERT tokenizer.....	32
Tabel 3. 10 Contoh Part-of-Speech Tagging.....	34
Tabel 3. 11 Contoh Chunking	34
Tabel 3. 12 Konfigurasi Hyperparameter.....	38
Tabel 3. 13 Contoh Tabel Hasil Pengujian Identifikasi	38
Tabel 4. 1 Penggalan Kode Proses Text Preprocessing	40
Tabel 4. 2 Penggalan Kode Proses Klasifikasi.....	41
Tabel 4. 3 Penggalan Kode Lapisan Bert Tokenizer.....	42
Tabel 4. 4 Penggalan Kode Lapisan Bert Model	43
Tabel 4. 5 Penggalan Kode Lapisan Bert Classifier	44
Tabel 4. 6 Penggalan Kode Proses POS Tagging	45
Tabel 4. 7 Penggalan Kode Proses Pembuatan Model POS Tagging.....	46
Tabel 4. 8 Penggalan Kode Proses Chunking	47
Tabel 4. 9 Penggalan Kode Proses Distributional Semantic Based Approach	48
Tabel 4. 10 Penggalan Kode Proses Pembuatan Similarity Model.....	49
Tabel 4. 11 Penggalan Kode Proses Validasi Truth Discovery	50
Tabel 4. 12 Penggalan Kode Membuat Tabel One Hot Encoding	50

Tabel 4. 13 Penggalan Kode Menghitung Claim Source Score (C_s).....	50
Tabel 4. 14 Penggalan Kode Proses Inisialisasi Claim Value	51
Tabel 4. 15 Penggalan Kode Menghitung Trustworthiness Score.....	51
Tabel 4. 16 Penggalan Kode Menghitung Belief Score.....	52
Tabel 4. 17 Penggalan Kode Proses Validasi Frasa Idiom	52
Tabel 4. 18 Hasil Validasi Model BERT untuk Klasifikasi Kalimat yang Mengandung Ekspresi Idiomatik	56
Tabel 4. 19 Hasil Evaluasi dengan Data Baru Model Klasifikasi BERT Terbaik	60
Tabel 4. 20 Hasil Evaluasi Identifikasi Ekspresi Idiomatik.....	61

DAFTAR GAMBAR

Gambar	Halaman
Gambar 2. 1 Konsep Chunking.....	8
Gambar 2. 2 Arsitektur BERT	10
Gambar 2. 3 Bert Embedding.....	11
Gambar 2. 4 Bert Attention.....	12
Gambar 2. 5 Bert Feed Forward Neural Network.....	13
Gambar 2. 6 Feed-Forward Neural Network	14
Gambar 2. 7 Bert Pooler.....	15
Gambar 2. 8 Bert Classifier.....	16
Gambar 2. 9 Metode Waterfall.....	21
Gambar 3. 1 Rancangan Antarmuka Halaman Beranda	26
Gambar 3. 2 Rancangan Antarmuka Halaman Identifikasi Idiom.....	27
Gambar 3. 3 Rancangan Antarmuka Halaman Kamus Idiom.....	27
Gambar 3. 4 Alur Umum Penelitian	30
Gambar 3. 5 Alur Proses Pembuatan Model Klasifikasi.....	32
Gambar 3. 6 Alur Proses Lapisan BERT Model.....	33
Gambar 3. 7 Alur Proses Identifikasi Ekspresi Idiomatik.....	35
Gambar 3. 8 Alur Proses Measuring Similarity	35
Gambar 3. 9 Alur Proses Idiom Phrase Selection	36
Gambar 3. 10 Alur Proses Truth Discovery	37
Gambar 4. 1 Halaman Beranda	53
Gambar 4. 2 Halaman Identifikasi Idiom.....	54
Gambar 4. 3 Halaman Kamus Idiom.....	54
Gambar 4. 4 Grafik Hasil Pengujian Klasifikasi Kalimat Epoch 3.....	57
Gambar 4. 5 Grafik Hasil Pengujian Klasifikasi Kalimat Epoch 4.....	57
Gambar 4. 6 Grafik Hasil Pengujian Klasifikasi Kalimat Epoch 5.....	58
Gambar 4. 7 Grafik Hasil Pengujian Klasifikasi Kalimat Epoch 6.....	58
Gambar 4. 8 Grafik Hasil Pengujian Klasifikasi Kalimat Epoch 7.....	59
Gambar 4. 9 Grafik Nilai Akurasi Pada Setiap Learning Rate dan Epoch	59

Gambar 4. 10 Grafik Hasil Evaluasi Identifikasi Ekspresi Idiomatik.....	62
---	----

DAFTAR LAMPIRAN

Lampiran	Halaman
Lampiran 1. Confusion Matrix untuk Pengujian Klasifikasi BERT Epoch 3.....	69
Lampiran 2. Confusion Matrix untuk Pengujian Klasifikasi BERT Epoch 4.....	70
Lampiran 3. Confusion Matrix untuk Pengujian Klasifikasi BERT Epoch 5.....	71
Lampiran 4. Confusion Matrix untuk Pengujian Klasifikasi BERT Epoch 6.....	72
Lampiran 5. Confusion Matrix untuk Pengujian Klasifikasi BERT Epoch 7.....	73
Lampiran 6. Confusion Matrix untuk Pengujian Identifikasi Truth Discovery	74

BAB I

PENDAHULUAN

1.1 Latar Belakang

Bahasa memegang peranan penting yaitu sebagai alat komunikasi dalam kehidupan sosial masyarakat. Dalam berbahasa, suatu makna tidak hanya dilambangkan dalam satu bentuk bahasa, tetapi juga dapat diungkapkan dalam berbagai bentuk. Bentuk adalah ekspresi makna, sehingga bentuk itu sendiri dapat merangsang penafsiran lebih dari satu makna, salah satu contohnya yaitu penggunaan idiom. Idiom biasa digunakan dalam kegiatan berkomunikasi sehari-hari baik secara tulisan maupun lisan. Penggunaan idiom ini sengaja dilakukan terutama untuk menyatakan sesuatu secara tidak langsung kepada lawan bicara, tanpa adanya kesalahan persepsi antara penutur dan petutur. Misalnya, kata pencuri lebih halus kedengarannya bila menggunakan kata panjang tangan. Penggunaan idiom sering ditemukan dalam puisi, novel, lirik lagu, surat kabar, majalah ataupun artikel (Virdaus, 2020).

Idiom adalah frasa yang terdiri dari urutan dua kata atau lebih yang memiliki makna yang tidak dapat diprediksi dari makna kata-kata individu atau mode kombinasi normalnya. Idiom ada di hampir semua bahasa dan sulit untuk diekstrak karena tidak ada algoritma yang dapat secara tepat menguraikan struktur sebuah idiom. Identifikasi ekspresi idiomatik adalah masalah yang menantang dengan penerapan yang luas. Mengidentifikasi ekspresi idiomatik sangat penting untuk aplikasi pemrosesan bahasa alami seperti *machine translation*, *information retrieval* dan sebagainya (Barrera *et al.*, 2011). Sebagian besar sistem mesin terjemahan berbasis aturan umumnya menerjemahkan ekspresi idiomatik dengan cara menerjemahkan kata demi kata penyusun ekspresi idiomatik, sehingga hasil terjemahan tidak menghasilkan makna yang sebenarnya dari ekspresi idiomatik tersebut.

Penelitian tentang identifikasi ekspresi idiomatik bahasa Indonesia belum pernah dilakukan sebelumnya. Namun terdapat beberapa penelitian yang serupa dalam bahasa lain. Seperti penelitian yang dilakukan Zayed, *et al.* (2018), pada

penelitian ini memperkenalkan pendekatan *semi-supervised* yang menggunakan representasi terdistribusi dari makna kata untuk menangkap metaforitas. Model *word embedding* digunakan untuk mengukur kemiripan semantik antara frasa kandidat dan kumpulan metafora yang telah ditentukan sebelumnya. Penelitian ini memperoleh nilai *precision* 0,5945, *recall* 0,756, *F-score* 0,6657 dan *accuracy* 0,6290. Kemudian terdapat penelitian yang dilakukan oleh Shutova *et al.* (2016), pada penelitian ini memperkenalkan metode identifikasi metafora pertama yang mengintegrasikan representasi makna yang dipelajari dari data linguistik dan visual dengan menerapkan metode *embedding* kata atau frasa untuk tugas identifikasi metafora. Pada penelitian ini memperoleh nilai *precision* yaitu 0,73, *recall* yaitu 0,80 dan *F-score* 0,76 dengan menggunakan dataset dari penelitian Tsvetkov *et al.* (2014) untuk metode *WordCos* menggunakan *linguistic embeddings*.

Berdasarkan permasalahan di atas, penulis mencoba melakukan penelitian mengenai identifikasi ekspresi idiomatik bahasa Indonesia pada suatu kalimat. Pertama-tama penulis akan melakukan proses klasifikasi kalimat menggunakan BERT untuk mengetahui apakah kalimat mengandung ekspresi idiomatik atau tidak. Selanjutnya ekspresi idiomatik diidentifikasi berdasarkan pendekatan berbasis semantik distribusi yang merupakan kombinasi dari pendekatan pada penelitian sebelumnya yang telah dipaparkan di atas. Metode ini mengidentifikasi ekspresi idiomatik pada tingkat frasa dilakukan dengan menjumlahkan kemiripan semantik antara kandidat dan kumpulan contoh ekspresi idiomatik dengan kemiripan semantik antara kata penyusun frasa kandidat. Kemudian melakukan validasi secara otomatis menggunakan kombinasi algoritma *Sums* dan *Average-Log* yang merupakan algoritma dari metode *Truth Discovery* dengan sumbernya merupakan berbagai macam website yang membahas mengenai ekspresi idiomatik bahasa Indonesia. Diharapkan dengan dilakukannya penelitian ini dapat membantu dalam mengidentifikasi penggunaan ekspresi idiomatik dalam suatu kalimat secara otomatis yang nantinya juga dapat dimanfaatkan untuk membantu mengoptimalkan aplikasi *machine translation* dalam menerjemahkan kalimat yang mengandung ekspresi idiomatik.

1.2 Rumusan Masalah

Berdasarkan latar belakang penelitian, dapat dirumuskan permasalahan pada penelitian ini yaitu:

- a. Bagaimana tingkat performa (*accuracy, precision, recall, F1-score*) klasifikasi kalimat berbahasa Indonesia yang mengandung ekspresi idiomatik atau tidak mengandung ekspresi idiomatik menggunakan BERT.
- b. Bagaimana tingkat performa (*accuracy, precision, recall, F1-score*) identifikasi ekspresi idiomatik pada kalimat berbahasa Indonesia menggunakan *Truth Discovery*.

1.3 Batasan Masalah

Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

- a. Data yang digunakan adalah kumpulan kalimat berpola dasar berbahasa Indonesia yang mengandung sebuah ekspresi idiomatik dan kalimat berpola dasar berbahasa Indonesia tidak mengandung ekspresi idiomatik.
- b. Sistem hanya dapat mengidentifikasi ekspresi idiomatik yang merupakan frasa nomina, verba dan adjektiva yang terdiri dari dua kata.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah, tujuan penelitian ini adalah sebagai berikut:

- a. Mengukur tingkat performa (*accuracy, precision, recall, F1-score*) klasifikasi kalimat berbahasa Indonesia yang mengandung ekspresi idiomatik atau tidak mengandung ekspresi idiomatik menggunakan BERT.
- b. Mengukur tingkat performa (*accuracy, precision, recall, F1-score*) identifikasi ekspresi idiomatik pada kalimat berbahasa Indonesia menggunakan *Truth Discovery*

1.5 Manfaat Penelitian

Adapun manfaat yang akan dicapai pada penelitian ini adalah sebagai berikut:

- a. Dapat dimanfaatkan untuk mengidentifikasi ekspresi idiomatik dari suatu kalimat.
- b. Dapat dimanfaatkan untuk mengoptimalkan aplikasi *machine translation* dalam menerjemahkan kalimat yang mengandung ekspresi idiomatik.

1.6 Sistematika Penulisan

Adapun sistematika penulisan yang digunakan dalam penelitian ini terbagi menjadi beberapa bab yang dapat dilihat pada penjelasan sebagai berikut.

BAB I PENDAHULUAN; bab ini memuat latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan pada penelitian identifikasi ekspresi idiomatik dan validasinya menggunakan *Truth Discovery*.

BAB II TINJAUAN PUSTAKA; pada bab ini terdapat ulasan teori untuk mendukung pendekatan pemecahan masalah (tinjauan teori) serta telaah hasil penelitian-penelitian terdahulu (tinjauan empiris) yang berkaitan dengan penelitian yang dilakukan mencakup ekspresi idiomatik bahasa Indonesia, klasifikasi menggunakan BERT, POS *tagging*, *chunking*, identifikasi ekspresi idiomatik menggunakan *Distributional Semantic Based Approach*, dan validasinya menggunakan *Truth Discovery*.

BAB III ANALISIS DAN PERANCANGAN SISTEM; bab ini penulis mengemukakan data yang digunakan serta metode yang digunakan dalam pengumpulan data, desain sistem yang dibuat serta desain evaluasi sistem yang akan digunakan pada penelitian identifikasi ekspresi idiomatik dan validasinya menggunakan *Truth Discovery*.

BAB IV HASIL DAN PEMBAHASAN; pada bab ini, penulis menguraikan implementasi sistem serta hasil penelitian dan analisis yang diperoleh pada penelitian identifikasi ekspresi idiomatik dan validasinya menggunakan *Truth Discovery*.

BAB V SIMPULAN DAN SARAN; bab ini memuat kesimpulan dari hasil penelitian yang telah dilakukan serta saran dari penulis tentang kemungkinan pengembangan dan pemanfaatan hasil penelitian lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Teori

2.1.1 Kelas Kata Bahasa Indonesia

Kelas kata merupakan kategori kata atau leksikal dalam satuan bahasa berdasarkan bentuk, fungsi, dan makna dalam sistem gramatikal (Moeliono *et al*, 2017:31). Kelas kata Bahasa Indonesia yang digunakan pada penelitian ini yaitu *Part of Speech Tagset* Bahasa Indonesia yang dirancang oleh Dinakaramani *et al*. (2014).

Tabel 2. 1 *Part of Speech Tagset* Bahasa Indonesia

No.	Tag	Deskripsi	Contoh
1	CC	<i>Coordinating conjunction</i>	dan, tetapi, atau
2	CD	<i>Cardinal number</i>	dua, 7916, sepertiga, 0,025, ribuan
3	OD	<i>Ordinal number</i>	ketiga, ke-4, pertama
4	DT	<i>Determiner / article</i>	para, sang, si
5	FW	<i>Foreign word</i>	<i>terms, conditions</i>
6	IN	<i>Preposition</i>	dalam, di, ke, oleh, pada, untuk
7	JJ	<i>Adjective</i>	bersih, panjang, hitam,
8	MD	<i>Modal and auxiliary verb</i>	boleh, harus, sudah
9	NEG	<i>Negation</i>	tidak, belum, jangan
10	NN	<i>Noun</i>	monyet, meja, rupiah
11	NNP	<i>Proper noun</i>	Boediono, Indonesia, BBKP, Januari
12	NND	<i>Classifier, partitive and measurement noun</i>	orang, ton, helai, lembar
13	PR	<i>Demonstrative pronoun</i>	ini, itu, sini, situ
14	PRP	<i>Personal pronoun</i>	saya, kami, kamu, dia, mereka
15	RB	<i>Adverb</i>	sangat, hanya, justru
16	RP	<i>Particle</i>	pun, -lah, -kah
17	SC	<i>Subordinating conjunction</i>	sejak, jika, seandainya, supaya, meski
18	SYM	<i>Symbol</i>	IDR, +, %, @
19	UH	<i>Interjection</i>	oh, ooh, aduh, ayo
20	VB	<i>Verb</i>	mengatur, pergi, bekerja
21	WH	<i>Question</i>	siapa, apa, kenapa, kapan, dimana, bagaimana, berapa
22	X	<i>Unknown word</i>	statemen
23	Z	<i>Punctuation</i>	. ! ? : ; () “ ‘

2.1.2 Ekspresi Idiomatik Bahasa Indonesia

Ekspresi idiomatik merupakan ungkapan yang maknanya tidak sesuai dengan prinsip komposisionalitas, dan tidak terkait dengan makna bagian (Chaer, 2013). Makna idiomatik adalah makna sebuah satuan bahasa yang menyimpang dari makna leksikal atau makna unsur-unsur pembentuknya. Hal ini berarti suatu idiom tidak dapat diterjemahkan kata per kata tetapi harus dilihat secara utuh dari unsur-unsur pembentuknya (Astuti, 2013). Ekspresi idiomatik yang akan diidentifikasi dalam penelitian ini yaitu idiom yang memiliki kategori frasa nomina, frasa verba dan frasa adjektiva yang disusun oleh dua kata. Contoh dari ekspresi idiomatik tersebut adalah sebagai berikut:

Tabel 2. 2 Contoh Ekspresi Idiomatik Bahasa Indonesia

Kategori		Contoh
Frasa Nomina	NN + NN	Kutu buku
	NN + JJ	Kuda hitam
	NN + VB	Bunga tidur
	CD + NN	Empat mata
Frasa Verba	VB + NN	Adu mulut Angkat kaki
	VB + JJ	Naik pitam
	VB + VB	Jatuh bangun
	VB + CD	Bermuka dua
Frasa Adjektiva	JJ + NN	Rendah hati Ringan tangan
	JJ + JJ	Panjang lebar

2.1.3 Text Preprocessing

Text preprocessing adalah proses yang terdiri dari serangkaian langkah untuk menyatukan, membersihkan dan menstandarisasikan data tekstual ke dalam bentuk yang dapat digunakan baik itu oleh NLP lain dan sistem cerdas yang didukung oleh *machine learning* dan *deep learning*. Teknik umum untuk *preprocessing* termasuk *tokenizing*, *removing special characters (punctuation)*, *case conversion*, *correcting spellings*, *removing stopwords*, *stemming*, dan *lemmatization* (Sarkar, 2019). Pada penelitian ini, sebelum data mentah dapat digunakan dalam melakukan identifikasi harus dilakukan *preprocessing* agar data

tersebut siap dan dapat diolah untuk tahap selanjutnya. Adapun tahapan *text preprocessing* yang dilakukan antara lain:

a. *Punctuation Removal*

Punctuation removal adalah proses untuk menghilangkan simbol-simbol yang terdapat pada teks.

b. *Tokenization*

Tokenization adalah proses memecah atau memisahkan data tekstual menjadi komponen yang lebih kecil dan lebih bermakna yang disebut *token*. *Token* dapat berupa karakter, kata, kalimat, maupun paragraf tergantung dengan kebutuhan penelitian.

c. *Case Conversion*

Case Conversion adalah tahapan mengkonversi bentuk huruf dalam teks menjadi seragam baik itu menjadi huruf kecil atau huruf besar. Pada penelitian ini semua karakter diubah menjadi huruf kecil.

2.1.4 *Part-of-Speech (POS) Tagging*

Part-of-Speech (POS) tagging adalah proses pengklasifikasian dan pelabelan pada setiap kata dalam kalimat dengan *Part-of-Speech* atau kelas kata yang sesuai untuk kata tersebut. *Tag* POS digunakan untuk membuat anotasi kata dan menggambarkan POS-nya, yang dapat dimanfaatkan dalam aplikasi berbasis NLP (Sarkar, 2019:163). *Part-of-speech tagging* merupakan komponen penting dalam penguraian sintaksis. *Part-of-Speech* adalah fitur yang berguna untuk menemukan entitas bernama seperti orang atau organisasi dalam teks dan tugas ekstraksi informasi lainnya. (Jurafsky dan Martin, 2017).

Pada penelitian ini, *part-of-speech tagging* dilakukan dengan menerapkan salah satu algoritma POS tagger dengan pendekatan *probabilistic-based* yaitu algoritma *Hidden Markov Model* (HMM). HMM merupakan pengembangan model statistik dari Markov Model. Adapun persamaan HMM ditunjukkan pada persamaan (2.1):

$$t_1^n = \prod_{i=1}^n \overbrace{P(w_i|t_i)}^{emisi} \overbrace{P(t_i|t_{i-1})}^{transisi} \quad (2.1)$$

Dengan $P(t_i|t_{i-1})$ merupakan probabilitas transisi yang mewakili probabilitas sebuah tag jika diketahui tag sebelumnya, yang dapat dihitung dengan persamaan (2.2):

$$P(t_i|t_{i-1}) = \frac{\text{Count}(t_{i-1}, t_i)}{\text{Count}(t_{i-1})} \quad (2.2)$$

Dan $P(w_i|t_i)$ merupakan probabilitas emisi yaitu probabilitas sebuah kata yang dilabeli *tag* tertentu, yang dihitung dengan persamaan (2.3).

$$P(w_i|t_i) = \frac{\text{Count}(t_i, w_i)}{\text{Count}(t_i)} \quad (2.3)$$

Dimana:

t_1^n = kelas kata yang dicari

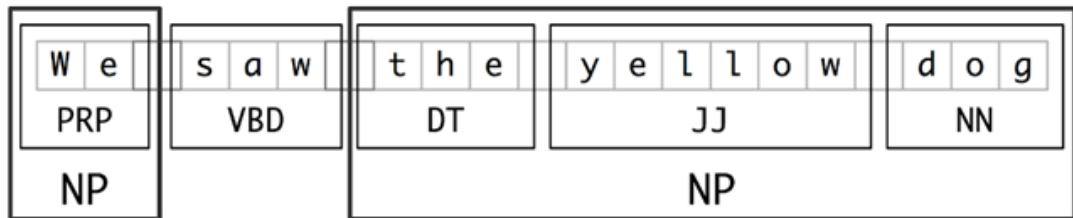
w_i = kata yang dicari kelas katanya

t_i = kelas kata dari w_i yang ada di *corpus*

t_{i-1} = kelas kata sebelum kelas kata dari w_i yang ada di *corpus*

2.1.5 *Chunking*

Chunking, atau biasa disebut juga dengan *Shallow Parsing*, adalah metode yang digunakan untuk membagi setiap kalimat menjadi beberapa segment yang tidak saling tumpang tindih. *Chunking* dapat dimanfaatkan untuk menemukan frasa kata benda, frasa kata kerja, frasa kata sifat, dan frase preposisional (Jurafsky dan Martin, 2017). Secara umum konsep *chunking* dapat dilihat pada gambar berikut:



Gambar 2. 1 Konsep *Chunking*

Pada penelitian ini, metode *chunking* digunakan untuk menemukan frasa nomina, frasa verba ataupun frasa adjektiva yang merupakan frasa kandidat yang sesuai dengan konstruksi sintaksis pembentuk ekspresi idiomatik. Untuk menemukan frasa kandidat tersebut pada suatu kalimat, pertama-tama penulis akan mendefinisikan tata bahasa *chunk* (*chunk grammar*) menggunakan *tag* POS, yang

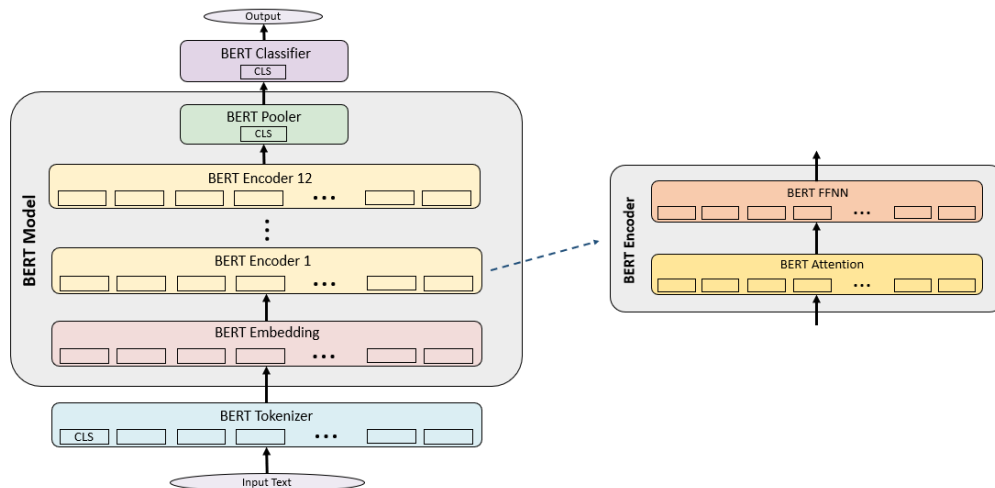
terdiri dari aturan *Regular Expressions* yang menunjukkan bagaimana kalimat harus dipotong.

2.1.6 *Bidirectional Encoder Representations from Transformers (BERT)*

Bidirectional Encoder Representations from Transformers atau disingkat BERT adalah model representasi bahasa terlatih yang dikembangkan oleh Devlin et al. (2019). BERT merupakan metode *state-of-the-art* dalam pembangunan *language model* dengan pendekatan *deep learning*. BERT menggunakan *transformer* yaitu sebuah mekanisme yang mempelajari hubungan kontekstual antara kata-kata dalam teks (Vaswani et al., 2017). Secara garis besar, *transformer* mencakup dua mekanisme terpisah yaitu, sebuah *encoder* yang berfungsi untuk membaca teks masukan dan sebuah *decoder* yang memproduksi prediksi untuk pekerjaan tertentu. Dikarenakan tujuan dari BERT adalah untuk menghasilkan sebuah *language model*, hanya mekanisme *encoder* yang digunakan dalam BERT.

Pada implementasinya, terdapat dua ukuran model yang ada pada BERT, yaitu BERT_{BASE} dan BERT_{LARGE}. Kedua ukuran model BERT ini memiliki banyak lapisan *encoder* atau *Transformer Blocks*. BERT_{BASE} memiliki *encoder* dengan 12 *layers*, 12 *self-attentions heads* dan *hidden size* sebesar 768. Sedangkan BERT_{LARGE} terdapat 24 *layers*, 16 *self-attention heads* dan *hidden size* sebesar 1024. BERT_{BASE} dilatih selama 4 hari menggunakan 4 *cloud TPU*s sedangkan BERT_{LARGE} membutuhkan 4 hari menggunakan 16 TPU_s.

Arsitektur model BERT berupa *multi-layer bidirectional transformer*, dimana *encoder* dari *transformer* membaca seluruh urutan kata sekaligus. Karakteristik *bidirectional* dari BERT memungkinkan model untuk mempelajari konteks dari sebuah kata berdasarkan lingkungan. Arsitektur BERT dapat dilihat pada gambar berikut (Kachuee dan Sharifkhani, 2022).



Gambar 2. 2 Arsitektur BERT

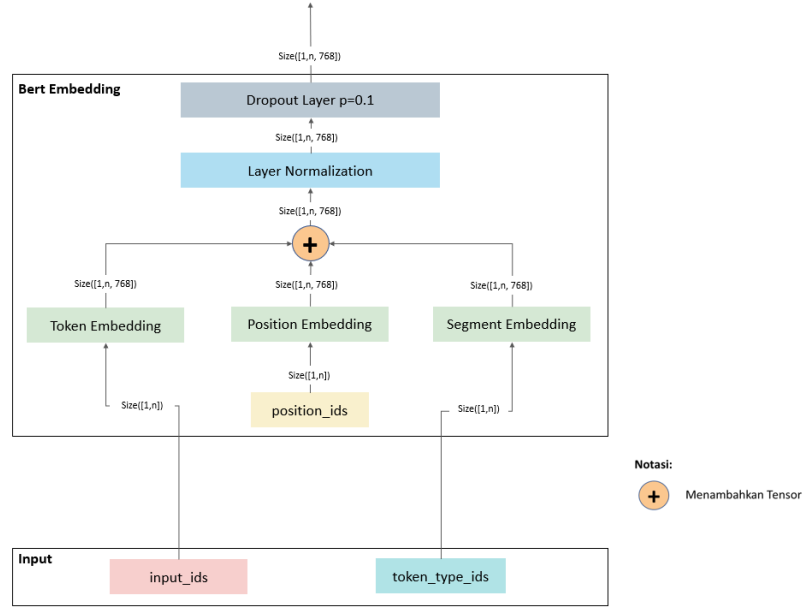
Pada Gambar 2.2 dapat dilihat arsitektur BERT terdiri dari beberapa bagian tahapan sebagai berikut:

1. BERT *Tokenizer*

Pada BERT *Tokenizer*, setiap kalimat ditokenisasi menjadi per kata atau sub kata menggunakan WordPiece. Selanjutnya setiap kalimat inputan akan diberikan token-token khusus yaitu [CLS] di awal kalimat yang merupakan token khusus yang digunakan sebagai indikator dalam tugas klasifikasi, dan token [SEP] di akhir kalimat sebagai pemisah jika inputan lebih dari satu kalimat, selanjutnya menambahkan token [PAD] untuk menyesuaikan dengan maksimal panjang inputan yang ditetapkan (*max_length*). Output dari BERT *tokenizer* terdiri dari token berupa kata, *input_ids* berupa bilangan unik atau id dari setiap token, *attention_mask* berupa bilangan biner yang menunjukkan posisi indeks token [PAD] sehingga model tidak memperhatikannya dan *token_type_ids* berupa bilangan biner yang membedakan antara kalimat pertama dan kedua.

2. BERT *Embedding*.

Pada BERT *Embedding* akan merepresentasikan kalimat inputan ke dalam bentuk vektor yang kemudian menjadi inputan pada BERT *Encoder*. Arsitektur pada lapisan BERT Embedding dapat lihat pada Gambar 2.3 (Rothman, 2021:63).



Gambar 2. 3 Bert Embedding

Pada lapisan ini terdapat tiga *embedding* yaitu (Devlin *et al.*, 2019):

- Token embedding* adalah representasi vektor dari tiap token (kata) pada kalimat. *Token embedding* menerima inputan berupa *input_ids* yaitu id unik dari setiap token.
- Segment embedding* adalah representasi untuk menunjukkan kalimat pertama atau kalimat kedua. *Segment embedding* menerima inputan berupa *token_type_ids* ini hanya memiliki dua representasi: 0 untuk token yang termasuk dalam kalimat pertama, dan 1 untuk token yang termasuk dalam kalimat kedua.
- Positional embedding* adalah representasi untuk urutan atau posisi setiap token dalam kalimat. *Positional embedding* menerima inputan berupa *position_ids* yaitu posisi dari setiap token pada kalimat. Berikut merupakan persamaan untuk menghitung *positional embedding* (PE) (Rothman, 2021:12):

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (2.4)$$

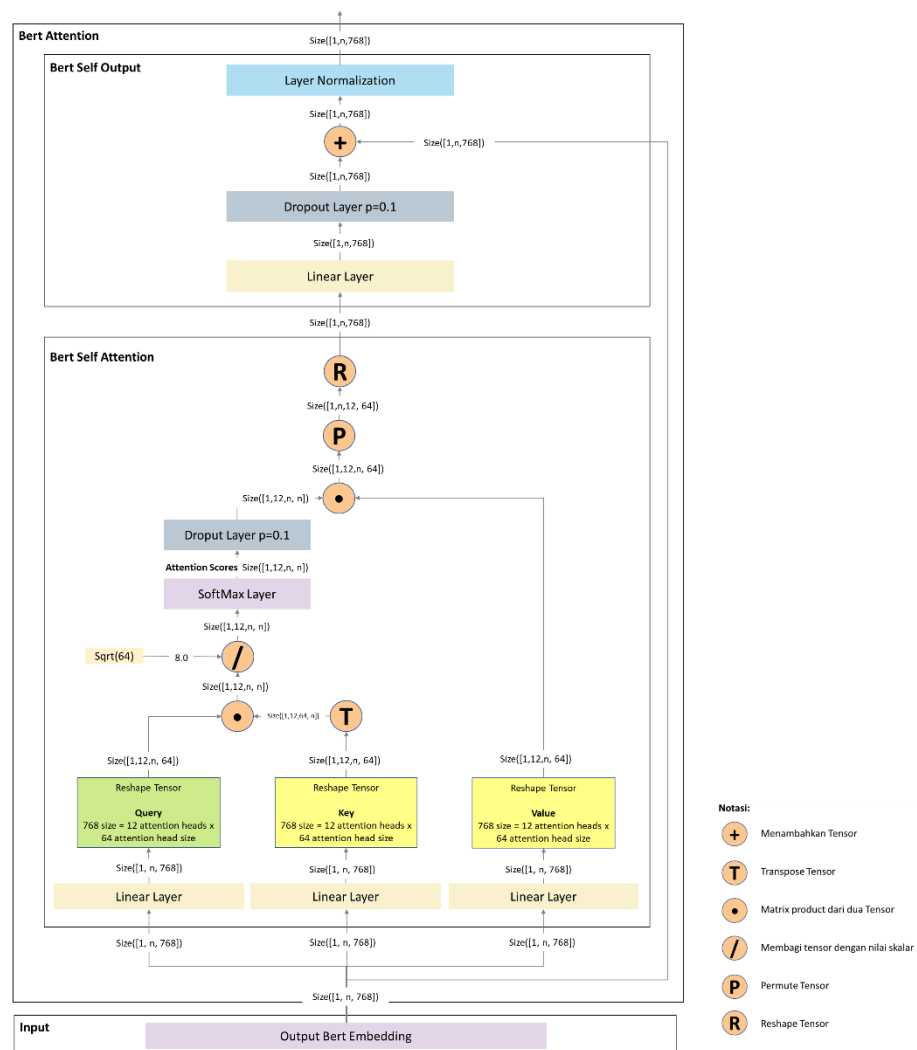
$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (2.5)$$

Dimana:

i = indeks vektor
 pos = posisi token
 d_{model} = dimensi dari input *embedding*

3. BERT Attention.

Self-attention merupakan mekanisme *attention* yang menghubungkan posisi yang berbeda dari *single* sekuensial untuk menghitung representasi dari sekuensial yang sama. Mekanisme *attention* memungkinkan *output* untuk memusatkan perhatian pada *input* sambil menghasilkan *output* sedangkan model *self-attention* memungkinkan *input* untuk berinteraksi satu sama lain. Arsitektur pada lapisan BERT Attention ditunjukkan pada Gambar 2.4 (Rothman, 2021:63).



Gambar 2. 4 Bert Attention

Self-attention membantu menemukan *contextual information* dari kata inputan dalam sekuensial. Adapun persamaan untuk menghitung output *self-attention* ditunjukkan oleh persamaan berikut:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.6)$$

Dengan nilai Q (*query vector*), K (*key vector*) dan V (*value vector*) dapat didapat dari persamaan berikut ini:

$$Q = X \times W^Q \quad (2.7)$$

$$K = X \times W^K \quad (2.8)$$

$$V = X \times W^V \quad (2.9)$$

Dimana:

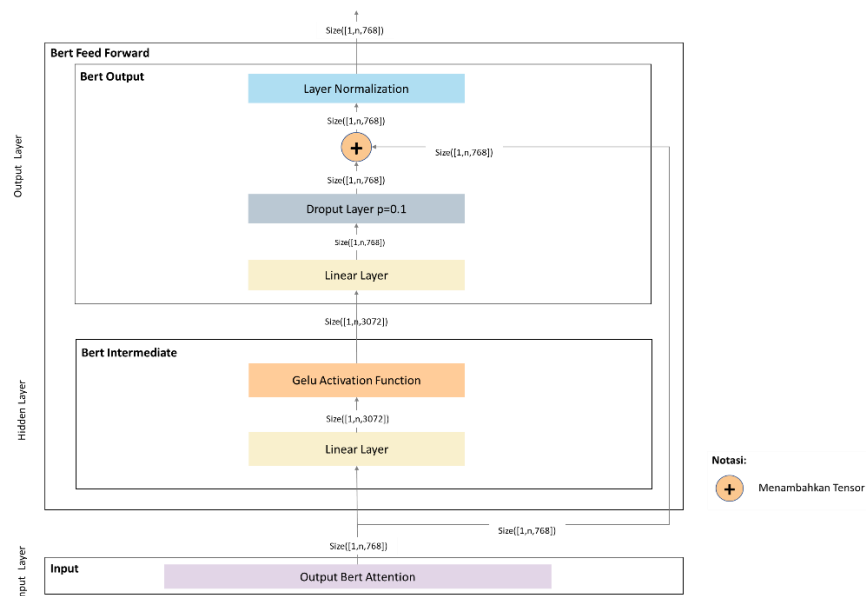
X = input *embedding vector*

W = *weight/bobot*

d_k = dimensi dari *key vector* yaitu 64

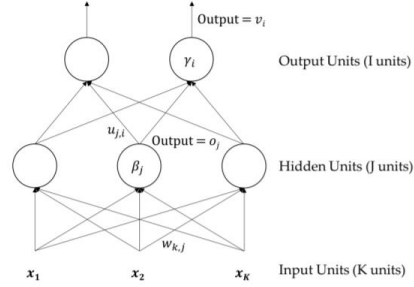
4. BERT Feed Forward Neural Network

Feed-forward neural network adalah jaringan di mana unit-unit terhubung tanpa siklus dan outputnya dikembalikan ke lapisan bawah. Arsitektur lapisan BERT *Feed Forward Neural Network* ditunjukkan pda Gambar 2.5 (Rothman, 2021:64).



Gambar 2. 5 Bert Feed Forward Neural Network

Secara umum *feed-forward neural network* memiliki tiga *layer* yaitu *input*, *hidden* dan *output layer*. *Output* dari *input layer* akan diterima sebagai *input* bagi *hidden layer*. Begitu seterusnya *hidden layer* akan mengirimkan hasilnya untuk *output layer*. Kegiatan ini dinamakan *feed forward*. *Feed-forward neural network* diilustrasikan pada gambar 2.6.



Gambar 2. 6 *Feed-Forward Neural Network*

Secara matematis, *feed-forward neural network* dapat dinotasikan sebagai persamaan berikut:

$$o_j = GELU \left(\sum_{k=1}^K x_k w_{k,j} + \beta_j \right) \quad (2.10)$$

$$v_i = GELU \left(\sum_{j=1}^J o_j u_{j,i} + \gamma_i \right) \quad (2.11)$$

Dimana:

x = vektor input

w, u = bobot pada *input layer* dan *hidden layer*

β, γ = bias pada *hidden layer* dan *output layer*

K, J = banyaknya *input unit* dan *hidden unit*

Kemudian untuk fungsi aktivasi yang digunakan yaitu GELU (*Gaussian Error Linear Unit*) merupakan fungsi aktivasi non-linear yang direpresentasikan sebagai berikut (Ravichandiran, 2021:77):

$$GELU(x) = xP(X \leq x) = x\Phi(x) \quad (2.12)$$

Dimana dapat diperkirakan dengan persamaan berikut untuk perhitungan yang lebih cepat:

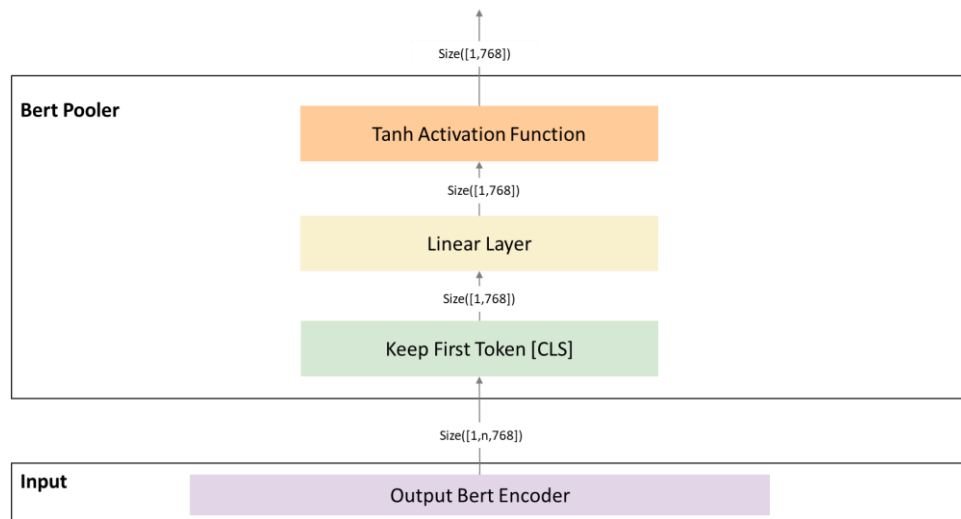
$$GELU(x) \approx 0.5x(1 + \tanh[\sqrt{2/\pi}(x + 0.044715x^3)]) \quad (2.13)$$

5. BERT Pooler

Pada bagian *pooler* akan mengambil output token pertama dari *Bert Encoder* yaitu token [CLS], token ini merupakan token klasifikasi khusus yang digunakan sebagai representasi urutan agregat untuk tugas klasifikasi. Setelah mendapatkan output token [CLS] selanjutnya akan melalui lapisan Linear. Selanjutnya untuk mendapatkan *pooled output* akan menerapkan fungsi aktivasi Tanh yang direpresentasikan sebagai berikut (Ravichandiran, 2021:104):

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.13)$$

Dimana e adalah eksponensial dan x adalah elemen dari input vektor. Arsitektur dari lapisan BERT Pooler dapat dilihat pada Gambar 2.7 (Kula *et al.*, 2021).

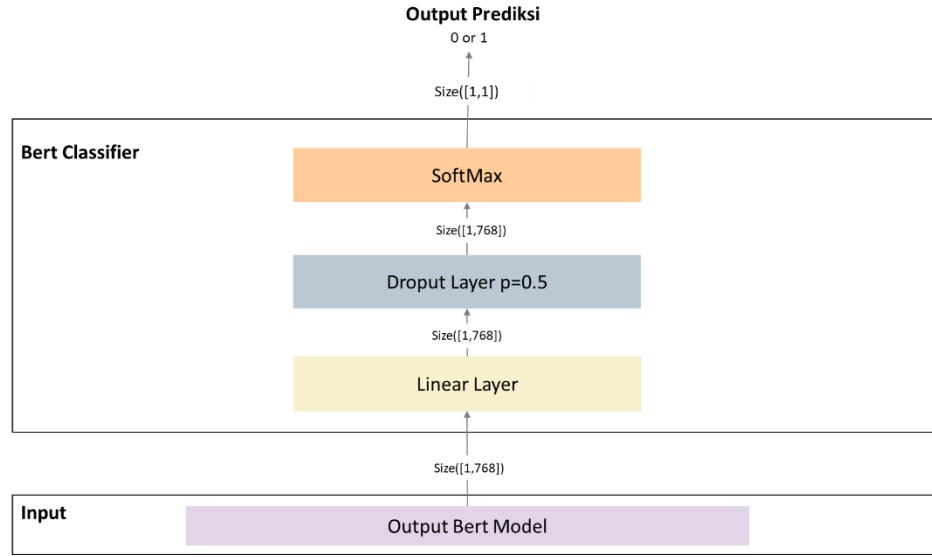


Gambar 2. 7 Bert Pooler

6. BERT Classifier

Output dari BERT Model yang digunakan untuk klasifikasi berasal vektor token [CLS] karena token ini dianggap melakukan pengumpulan rata-rata atas token kata untuk mendapatkan vektor dari kalimat. *Layer* terakhir pada BERT Classifier menghasilkan *logits*. *Logits* adalah *output* yang berupa prediksi probabilitas kasar dari kalimat yang akan diklasifikasikan.

Arsitektur dari lapisan BERT *Classifier* dapat dilihat pada Gambar 2.7 (Kula *et al.*, 2021).



Gambar 2. 8 *Bert Classifier*

Selanjutnya pada lapisan ini akan menggunakan fungsi *softmax* untuk mengubah *logits* tersebut menjadi probabilitas dengan mengambil eksponen dari tiap nilai *logit* sehingga total probabilitasnya adalah tepat 1. Sehingga nilai probabilitas akan berada di antara 0 atau angka positif (Munika *et al.*, 2019). Fungsi *softmax* ditunjukkan oleh persamaan berikut ini:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (2.14)$$

Dengan $i = 1, \dots, K$ dimana, $z = (z_1, \dots, z_K) \in \mathbb{R}^K$ adalah vektor input yang dimasukkan pada fungsi softmax atau output dari *layer* terakhir yang disebut juga dengan *logits*. Inputan yang dapat diterima merupakan bilangan real. e^{z_i} adalah eksponensial dari tiap elemen dari input vektor. $\sum_{j=1}^K e^{z_j}$ adalah proses normalisasi untuk memastikan semua nilai output dari softmax akan berjumlah tepat 1 dan masing-masing nilai berada di antara kisaran (0, 1).

2.1.7 *Distributional Semantic Based Approach*

Pada penelitian ini Identifikasi ekspresi idiomatik bahasa Indonesia yang dilakukan dengan menggunakan pendekatan berbasis semantik distribusi

(*Distributional Semantic Based Approach*). Semantik distribusional adalah pendekatan semantik yang didasarkan pada konteks kata-kata atau frasa dalam korpora besar untuk mengukur dan mengkategorikan kesamaan semantik antara item linguistik (kata-kata atau frasa) tersebut berdasarkan sifat distribusinya dimana item linguistik dengan distribusi serupa memiliki semantik yang sama.

Pada pendekatan semantik distribusional ini identifikasi ekspresi idiomatik dilakukan berdasarkan konteks kata-kata dan frasa pada data kumpulan kalimat yang digunakan untuk mengumpulkan informasi distribusi dalam vektor berdimensi tinggi, kemudian mendefinisikan kesamaan distribusi/semantik melalui kesamaan vektor. Selanjutnya untuk menentukan suatu frasa merupakan ekspresi idiomatik atau tidak dilakukan dengan mengkombinasikan metode pada penelitian yang dilakukan oleh Zayed, *et al.* (2018) dan penelitian yang dilakukan oleh Shutova *et al.* (2016) yaitu dengan menjumlahkan kemiripan semantik antara kandidat dan kumpulan contoh ekspresi idiomatik dengan kemiripan semantik antara kata penyusun frasa kandidat kemudian membandingkannya yang dapat dilihat pada persamaan (2.15):

$$sim = \sum_{i=1}^n sim(phrase, idiom\ example_i) + sim(w_1, w_2) \quad (2.15)$$

Dimana:

sim = similarity

$phrase$ = frasa kandidat idiom

$idiom\ example$ = contoh idiom

w_1, w_2 = kata-kata penyusun frasa kandidat idiom

Pendekatan ini menerapkan *Embedding* dari metode BERT untuk menghasilkan representasi vektor dari kata-kata dan juga frasa. Selanjutnya, pendekatan ini menggunakan *cosine similarity* untuk mengukur kesamaan semantik, yang ditunjukkan oleh persamaan (2.7):

$$\cos(u, v) = \frac{u \cdot v}{\|u\| \|v\|} = \frac{\sum_{i=1}^n u_i v_i}{\sqrt{\sum_{i=1}^n u_i^2} \sqrt{\sum_{i=1}^n v_i^2}} \quad (2.16)$$

Dimana:

u, v = vektor kandidat

u_i, v_i = bobot dari vektor u, v

2.1.8 *Truth Discovery*

Pertumbuhan data web yang cepat memberikan informasi yang sangat banyak, informasi ini tersedia dalam jumlah besar dan dalam berbagai format tidak terstruktur seperti Wikipedia, artikel berita, situs web perusahaan dan lain-lain. Meskipun informasi tersedia dari lebih banyak sumber, namun terkadang informasi yang diberikan belum tentu benar dan kadang saling bertentangan. Oleh karena itu terdapat metode *Truth Discovery* (Berti-Equille dan Borge-Holthoefer, 2015). *Truth Discovery* atau bisa disebut pengecekan fakta bertugas untuk menemukan pernyataan yang benar diantara banyaknya pernyataan yang diklaim yang diajukan banyak sumber untuk objek yang sama (Sanjaya et al, 2018). Terdapat beberapa algoritma dari *Truth Discovery* diantaranya adalah *Majority voting*, *TruthFinder*, *Accu*, *Sums*, *Average-Log*, *Investment*, *PooledInvestment*, *2-Estimates*, *3-Estimates*, *SimpleLCA*, *GuessLCA* dan *CRH* (Fang et al, 2017). Pada penelitian ini validasi ekspresi idiomatik yang didapat dari hasil identifikasi akan divalidasi menggunakan kombinasi algoritma *Sums* dan *Average-Log*.

Algoritma *Sums* merupakan algoritma yang terinspirasi *Hubs and Authorities* (Kleinberg, 1999) dengan memberi setiap halaman skor hub dan skor otoritas, di mana skor hubnya adalah jumlah otoritas halaman yang ditautkan dan otoritasnya adalah jumlah skor hub halaman yang terhubung dengannya. Ini disesuaikan dengan pencarian fakta dengan melihat sumber sebagai hub dan klaim sebagai otoritas (Pasternack dan Roth, 2010). Algoritma *Sums* dapat ditunjukkan oleh persamaan (2.17) dan (2.18):

$$T^i(s) = \sum_{c \in C_s} B^{i-1}(c) \quad (2.17)$$

$$B^i(c) = \sum_{s \in S_c} T^i(s) \quad (2.18)$$

Sums memungkinkan sumber untuk meningkatkan skor kepercayaan mereka hanya dengan membuat banyak klaim, yang berpotensi tidak diinginkan.

Jumlah klaim tidak boleh diabaikan, namun, sumber dengan akurasi 90% lebih dari 100 klaim pasti lebih dapat dipercaya daripada yang memiliki akurasi 90% dengan lebih dari 10 klaim. Berdasarkan hal tersebut, *Average-Log* mencoba mengatasi dengan menetapkan skor kepercayaan untuk sumber ke skor keyakinan rata-rata klaimnya, dikalikan dengan logaritma jumlah fakta yang diklaimnya (Pasternack dan Roth, 2010). Algoritma *Average-Log* dapat ditunjukkan oleh persamaan (2.19):

$$T^i(s) = \log|C_s| \cdot \frac{\sum_{c \in C_s} B^{i-1}(c)}{|C_s|} \quad (2.19)$$

Dimana:

- S = kumpulan sumber
- S_c = kumpulan sumber yang memberikan klaim c
- C = kumpulan klaim
- C_s = kumpulan klaim disediakan oleh sumber s
- $T^i(s)$ = *trustworthiness score* dari sumber s
- $B^i(c)$ = *belief score* dari klaim c
- $B^{i-1}(c)$ = *belief score* sebelumnya dari klaim c

2.1.9 Confusion Matrix

Confusion matrix salah satu cara paling populer untuk mengevaluasi model klasifikasi. *Confusion matrix* dapat digunakan untuk *binary classification* serta *multi-class classification model*. *Confusion matrix* dibuat dengan membandingkan label kelas yang diprediksi dari suatu titik data dengan label kelas yang sebenarnya. Perbandingan ini diulangi untuk seluruh dataset dan hasil perbandingan ini dikompilasi dalam format matriks atau tabel. Tabel berikut menunjukkan *Confusion matrix* untuk *binary classification* (Sarkar, 2019:310-314).

Tabel 2. 3 *Confusion Matrix*

		PREDICTED LABELS	
		n' (Predicted)	p' (Predicted)
TRUE LABELS	n (True)	TN	FP
	p (True)	FN	TP

Dari tabel tersebut dapat dihasilkan:

- a. *True Positive* (TP), yaitu total hasil dari prediksi kelas positif dan sesuai dengan kelas aslinya yang positif.
- b. *True Negative* (TN), yaitu total hasil dari prediksi kelas negatif dan sesuai dengan kelas aslinya yang negatif.
- c. *False Positive* (FP), yaitu total hasil dari prediksi kelas positif namun tidak sesuai dengan kelas aslinya yang negatif.
- d. *False Negative* (FN), yaitu total hasil dari prediksi kelas negatif namun tidak sesuai dengan kelas aslinya yang positif.

Pengukuran evaluasi sistem dapat digunakan dengan menggunakan empat parameter, yaitu *accuracy*, *precision*, *recall* dan *F1-Score*.

Accuracy merupakan perbandingan antara total hasil prediksi yang benar dengan total keseluruhan data.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.20)$$

Precision merupakan perbandingan antara total hasil prediksi kelas positif yang benar dengan total keseluruhan data yang diprediksi sebagai kelas positif.

$$Precision = \frac{TP}{TP + FP} \quad (2.21)$$

Recall merupakan perbandingan antara total hasil prediksi kelas positif yang benar dengan total keseluruhan data yang benar-benar positif.

$$Recall = \frac{TP}{TP + FN} \quad (2.22)$$

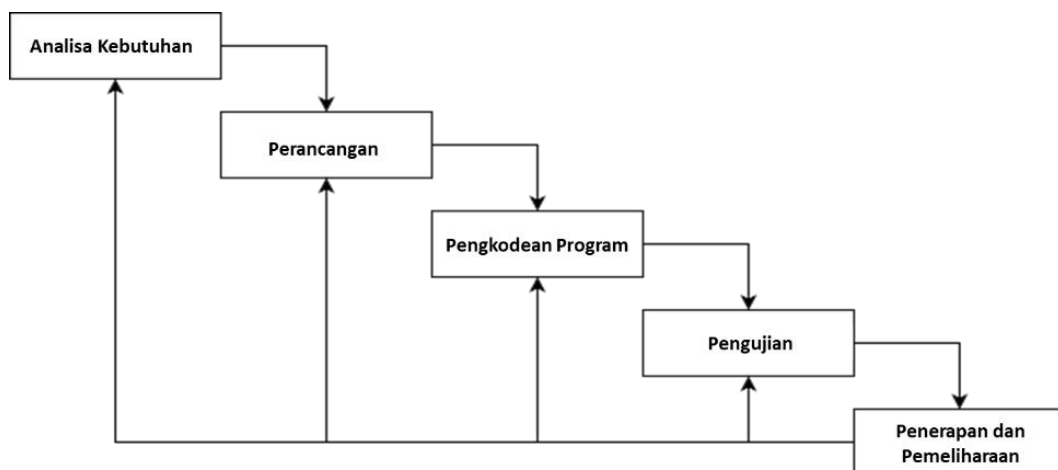
Setelah mendapat nilai *Recall* dan *Precision*, maka dilakukan perhitungan menggunakan *F1-score*. *F1-score* digunakan untuk mengukur kombinasi hasil *precision* dan *recall*, sehingga menjadi satu nilai pengukuran.

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2.23)$$

2.1.10 Metode Pengembangan Perangkat Lunak *Waterfall*

Metode pengembangan perangkat lunak dengan model *Waterfall* pertama kali diperkenalkan oleh Windows W. Royce pada tahun 1970. Menurut Pressman

(2012), model *Waterfall* merupakan suatu model pengembangan secara sekuensial. Model *Waterfall* bersifat sistematis dan berurutan dalam membangun sebuah perangkat lunak. Alur atau tahapan model *Waterfall* dapat dilihat pada gambar 2.9 berikut.



Gambar 2. 9 Metode *Waterfall*

Kelebihan model *Waterfall* yaitu merupakan model pengembangan paling andal dan paling lama digunakan. Model pengembangan *Waterfall* cocok digunakan untuk sistem yang bersifat generik dalam artian sistem dapat diidentifikasi semua kebutuhannya dari awal dengan spesifikasi yang umum. Model pengembangan ini sesuai untuk tugas akhir yang memiliki tujuan untuk membangun sebuah sistem dari awal dengan mengumpulkan kebutuhan sistem yang akan dibangun sesuai dengan topik penelitian yang dipilih sampai dengan produk tersebut diuji (Susanto & Andriana, 2016).

2.2 Tinjauan Empiris

Berikut ini beberapa penelitian sebelumnya yang dijadikan dasar atau perbandingan dalam melakukan penelitian yaitu:

1. Analisis Morfologi untuk Menangani *Out-of-Vocabulary Words* pada *Part-of-Speech Tagger* Bahasa Indonesia Menggunakan *Hidden Markov Model* (Ramadhanti *et al*, 2019)

Pada penelitian ini, peneliti membandingkan POS tagger yang menggunakan HMM+AM (analisis morfologi) dan POS tagger HMM tanpa

AM, dengan menggunakan *train corpus* dan *testing corpus* yang sama. Hasil yang diperoleh dari sistem dengan menggunakan HMM saja memiliki akurasi 97.54%, sedangkan sistem yang menggunakan kombinasi HMM dengan metode analisis morfologi memiliki akurasi tertinggi 99.14%. Perbedaan dengan penelitian yang akan dilakukan penulis, yaitu pada penelitian penulis hanya menggunakan metode HMM untuk melakukan POS *tagging*.

2. Aplikasi Penentuan Kategori dan Fungsi Sintaksis Kalimat Bahasa Indonesia (Syahroni dan Harsono, 2019)

Pada penelitian ini, peneliti menggunakan teknik *chunking* untuk memecah kalimat menjadi potongan kata atau frasa dengan membuat *chunk grammar* yang berupa aturan yang didefinisikan menggunakan *Regular Expression* yang akan mengindikasikan cara kalimat tersebut dipecah menjadi potongan frasa. Frasa yang dibentuk antara lain Frasa Nominal (NP), Frasa Verbal (VP), Frasa Adjektiva (AP), Frasa Adverbial (ADVP), Frasa Preposisional (PP) dan Frase Numeralia (NUMP). Pada penelitian ini mendapatkan akurasi sebesar 93,32 %. Persamaan dengan penelitian yang akan dilakukan penulis, yaitu menggunakan *teknik chunking* untuk memecah kalimat menjadi potongan frasa dengan membuat *chunk grammar* yang didefinisikan menggunakan *Regular Expression*.

3. *Fine-grained Sentiment Classification using BERT* (Munika et al, 2019)

Pada penelitian ini, peneliti menggunakan BERT untuk melakukan klasifikasi sentimen pada dataset SST (*Stanford Sentiment Treebank*). Peneliti menggunakan dua tipe dataset SST yaitu SST-2 dengan dua label (positif dan negatif) dan SST-5 dengan lima label (sangat negatif, negatif, netral, positif dan sangat positif). Pada penelitian ini memperoleh akurasi sebesar 94,0 pada dataset SST-2 dan 83,9 pada dataset SST-5. Persamaan dengan penelitian yang akan dilakukan penulis, yaitu menggunakan BERT untuk melakukan tugas klasifikasi, dimana pada penelitian penulis akan dilakukan klasifikasi kalimat yang mengandung idiom dan kalimat yang tidak mengandung idiom.

4. *Phrase-Level Metaphor Identification using Distributed Representations of Word Meaning* (Zayed *et al*, 2018)

Pada penelitian ini memperkenalkan pendekatan *semi-supervised* yang menggunakan representasi distribusi makna kata untuk menangkap metaforitas. Pada penelitian ini fokus untuk mengidentifikasi pasangan kata kerja-kata benda di mana kata kerja digunakan secara metaforis. Peneliti mengekstrak hubungan tata bahasa kata kerja-kata benda menggunakan *Stanford parser*. Peneliti kemudian menggunakan model *word embeddings* untuk mengukur kemiripan semantik antara kandidat dan kumpulan metafora yang telah ditentukan sebelumnya. Penelitian ini memperoleh nilai *precision* 0,5945, *recall* 0,756, *F-score* 0,6657 dan *accuracy* adalah 0,6290.

Persamaan dengan penelitian yang akan dilakukan penulis, yaitu mengidentifikasi metafora atau idiom dengan mengukur kemiripan semantik antara kandidat dan kumpulan metafora yang telah ditentukan sebelumnya dengan menerapkan model *word embeddings*. Perbedaan dengan penelitian yang akan dilakukan penulis adalah penelitian ini hanya fokus untuk mengidentifikasi pasangan kata kerja-kata benda.

5. *Black Holes and White Rabbits: Metaphor Identification with Visual Features* (Shutova *et al*, 2016)

Pada penelitian ini memperkenalkan metode identifikasi metafora pertama yang mengintegrasikan representasi makna yang dipelajari dari data linguistik dan visual. Peneliti membangun representasi menggunakan model *skip-gram* Mikolov *et al.* (2013) dilatih dengan data tekstual untuk mendapatkan *linguistic embeddings*. Fokus percobaan peneliti adalah pada ekspresi metafora dalam subjek kata kerja, kata kerja-objek langsung dan pengubah kata sifat-konstruksi kata benda. Peneliti menggunakan satu set operasi aritmatika pada vektor *embedding* kata dan frasa untuk mengklasifikasikan frasa sebagai literal atau metaforis. Pada penelitian ini memperoleh nilai *precision* yaitu 0,73, *recall* yaitu 0,80 dan *F-score* 0,76 pada Tsvetkov *et al.* dataset untuk metode *WordCos* menggunakan

linguistic embeddings. Persamaan dengan penelitian yang akan dilakukan penulis, yaitu menerapkan *embeddings* kata atau frasa untuk tugas identifikasi metafora.

6. *Knowing What to Believe (when you already know something)* (Pasternack dan Roth, 2010)

Pada penelitian ini memperkenalkan beberapa algoritma untuk *truth discovery* yaitu *Sums*, *Average-Log*, *Investment*, *PooledInvestment* yang mengekspresikan penalaran secara general yaitu *common-sense reasoning* dan fakta spesifik yang sudah diketahui pengguna sebagai *first-order logic* kemudian menerjemahkannya ke dalam *tractable linear program*. Pada penelitian ini menghasilkan nilai akurasi untuk setiap algoritma yaitu *Sums* 89,53%, *Average-Log* 89,24%, *Investment* 88,34%, dan *PooledInvestment* 90.01% untuk *Basic Biographies* dataset.

Dari penelitian yang dilakukan oleh Pasternack dan Roth (2010), penulis akan menerakan kombinasi algoritma yang diperkenalkan dalam penelitian ini, yaitu algoritma *Sums* dan *Average-Log* untuk melakukan validasi informasi.

BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Metode Pengembangan Sistem

Pada penelitian ini, sistem identifikasi ekspresi idiomatik berbasis web dengan menggunakan metode pengembangan dengan model *Waterfall*. Adapun tahapan dari proses pengembangan sistem identifikasi ekspresi idiomatik dengan model *Waterfall* adalah sebagai berikut.

3.1.1 Analisa Kebutuhan

Pada tahap ini dilakukan pengumpulan mengenai spesifikasi kebutuhan sistem yang akan dibuat meliputi kebutuhan fungsional dan kebutuhan non-fungsional. Adapun kebutuhan fungsional sistem dapat dilihat pada Tabel 3.1.

Tabel 3. 1 Kebutuhan Fungsional Sistem

No	Kebutuhan Fungsional
1	Sistem dapat menerima masukan berupa kalimat
2	Sistem dapat mengklasifikasikan kalimat masukan sebagai kalimat yang mengandung ekspresi idiomatik atau kalimat yang tidak mengandung ekspresi idiomatik
3	Sistem dapat melakukan identifikasi ekspresi idiomatik pada kalimat masukan
4	Sistem dapat menampilkan arti dari idiom yang teridentifikasi dari kalimat masukan
5	Sistem dapat menerima masukan berupa frasa ekspresi idiomatik
6	Sistem dapat menampilkan arti dan contoh penggunaan dari frasa ekspresi idiomatik yang di masukan

Selain kebutuhan fungsional, analisis kebutuhan non-fungsional juga perlu dilakukan untuk mengetahui spesifikasi kebutuhan sistem yaitu analisis perangkat keras dan analisis perangkat lunak dapat dilihat pada Tabel 3.2.

Tabel 3. 2 Kebutuhan Non-Fungsional Sistem

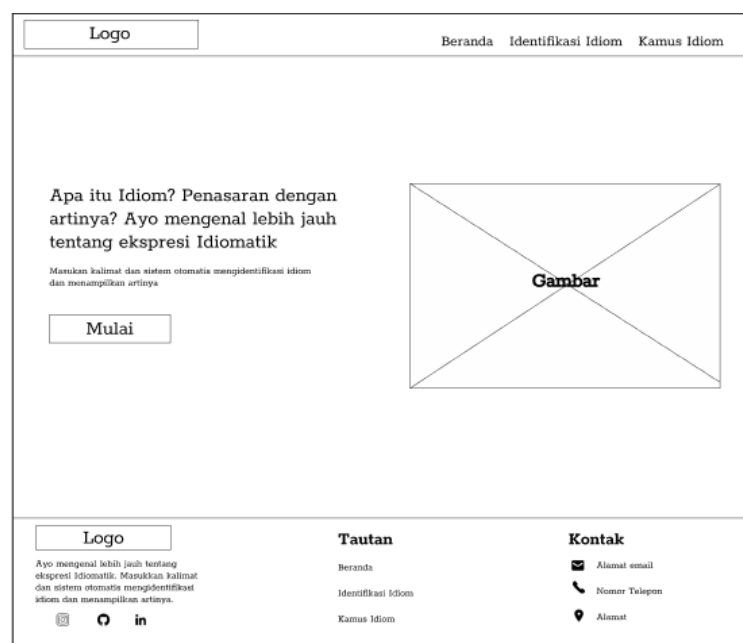
No	Kebutuhan Non-Fungsional
1	Analisis perangkat keras <ul style="list-style-type: none">• Processor Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz• Memory 8192MB RAM• SSD 475GB
2	Analisis perangkat lunak <ul style="list-style-type: none">• Sistem Operasi Windows 10 Home Single Language• Bahasa Pemrograman Python 3.7.11• Framework Flask 2.0.3• PyCharm Community Edition 2020.2.1• Google Collab

3.1.2 Perancangan Antarmuka Sistem

Pada tahap ini sistem yang dibangun akan direpresentasikan dengan desain antarmuka. Berikut merupakan perancangan antarmuka sistem identifikasi ekspresi idiomatik berbasis web yang dibangun.

1. Rancangan antarmuka halaman beranda

Pada halaman beranda menampilkan nama sistem serta deskripsi singkat dari sistem. Rancangan antarmuka dari halaman beranda sistem dapat dilihat pada Gambar 3.1.



Gambar 3. 1 Rancangan Antarmuka Halaman Beranda

2. Rancangan antarmuka halaman identifikasi idiom

Halaman identifikasi digunakan untuk melakukan identifikasi ekspresi idiomatik. Pada halaman ini terdapat *text box* untuk menerima inputan dari user yang berupa suatu kalimat berpola dasar berbahasa Indonesia. Kemudian terdapat tombol identifikasi untuk melakukan proses identifikasi ekspresi idiomatik pada kalimat inputan dan juga terdapat tombol cari arti untuk menampilkan arti dari ekspresi idiomatik yang telah diidentifikasi sebelumnya. Rancangan antarmuka dari halaman identifikasi dapat dilihat pada Gambar 3.2.

Gambar 3. 2 Rancangan Antarmuka Halaman Identifikasi Idiom

3. Rancangan halaman kamus idiom

Pada halaman kamus idiom terdapat *text box* untuk menerima inputan dari user yang berupa huruf, kata, atau frasa berbahasa Indonesia. Kemudian terdapat tombol cari untuk melakukan proses pencarian berdasarkan inputan dan menampilkan hasil berupa ekspresi idiomatik beserta arti dan contoh penggunaannya dalam kalimat. Rancangan antarmuka dari halaman kamus idiom dapat dilihat pada Gambar 3.3.

Kata Pembentuk	Ekspresi Idiomatik	Arti	Contoh Penggunaan
Kata pembentuk ekspresi idiomatik	Frasa ekspresi idiomatik	Arti dari ekspresi idiomatik	Contoh penggunaan dari ekspresi idiomatik pada kalimat

Gambar 3. 3 Rancangan Antarmuka Halaman Kamus Idiom

3.1.3 Pengkodean Program

Tahap ini merupakan tahap implementasi dari desain sistem yang telah dibuat ke dalam kode pemrograman komputer. Bahasa pemrograman yang digunakan adalah Python 3.7, HTML dan CSS dan proses pengkodean program dilakukan dengan menggunakan PyCharm dan Google Colab. Tahap pengkodean dapat dilihat lebih lanjut pada Bab 4.

3.1.4 Pengujian

Pada tahap ini sistem yang telah dibangun dilakukan pengujian baik dari aspek logika maupun fungsional, apakah sistem yang telah dibangun telah sesuai dengan yang direncanakan serta meminimalkan kesalahan (*error*) pada sistem sehingga diperoleh hasil yang baik. Skenario pengujian dapat dilihat pada subbab 3.4.1 sedangkan evaluasi sistem dapat dilihat pada Subbab 3.4.2. Hasil dari evaluasi sistem dapat dilihat pada Bab 4.

3.1.5 Penerapan dan Pemeliharaan

Dalam membangun sebuah sistem, tidak jarang terjadi kesalahan yang tidak ditemukan pada langkah sebelumnya serta penambahan fitur pada sistem setelah sistem itu dibuat sebagai kebutuhan baru. Tahapan ini dilakukan bertujuan untuk mengembangkan perubahan-perubahan dan memperbaiki kesalahan-kesalahan yang mungkin terjadi saat sistem sudah digunakan.

3.2 Data dan Metode Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah data sekunder yaitu data yang sudah tersedia sebelum peneliti memulai penelitian (Anggito *et al*, 2018). Terdapat tiga data yang digunakan, dimana data ini bersumber dari media internet, buku ataupun publikasi. Data pertama berupa *Indonesian Manually Tagged Corpus*, yaitu kumpulan kalimat bahasa Indonesia yang telah diberikan tag secara manual. Data ini digunakan pada tahap *POS Tagging*. Contoh data dapat dilihat pada Tabel 3.3.

Tabel 3. 3 *Indonesian Manually Tagged Corpus*

Contoh Kalimat dan Kelas Katanya
Belum/NEG lama/JJ ini/PR kawanan/NN monyet /NN menyerbu/VB kantor/NN perdana menteri/NN dan/CC departemen/NN pertahanan/NN ./Z
Jumlah/NN dan/CC harga/NN barang/NN itu/PR masih/MD dirundingkan/VB ./Z
Buaya-buaya/NN itu/PR berukuran/VB panjang/NN antara/IN 40/CD -/Z 50/CD centimeter/NNN ./Z

Data kedua berupa kumpulan kalimat berpola dasar berbahasa Indonesia yang mengandung sebuah ekspresi idiomatik dan kalimat berpola dasar berbahasa Indonesia yang tidak mengandung ekspresi idiomatik. Data ini berjumlah 2000 kalimat yang telah dilabeli sebagai kalimat biasa dan kalimat idiom secara manual oleh pakar dan juga berdasarkan pada buku kamus idiom bahasa Indonesia, dengan jumlah kalimat pada setiap label yaitu 1000 kalimat. Data ini digunakan pada tahap klasifikasi kalimat dan tahap identifikasi ekspresi idiomatik. Contoh data dapat dilihat pada Tabel 3.4.

Tabel 3. 4 Contoh Kalimat Bahasa Indonesia

indeks	kalimat	kategori	frasa_idiom	validasi
1	Orang tua itu rela membanting tulang demi menyekolahkan ketiga anaknya.	kalimat_idiom	membanting tulang	idiom
2	Ayah adalah tangan kanan Pak Camat.	kalimat_idiom	tangan kanan	idiom
3	Huda suka menjadikan Andik sebagai kambing hitam.	kalimat_idiom	kambing hitam	idiom
4	Gadis itu sedang membaca sebuah buku novel.	kalimat_biasa	none	bukan_idiom
5	Tangan kanan Roni terkena minyak panas saat menggoreng ikan.	kalimat_biasa	none	bukan_idiom
6	Ayah membeli kambing hitam.	kalimat_biasa	none	bukan_idiom

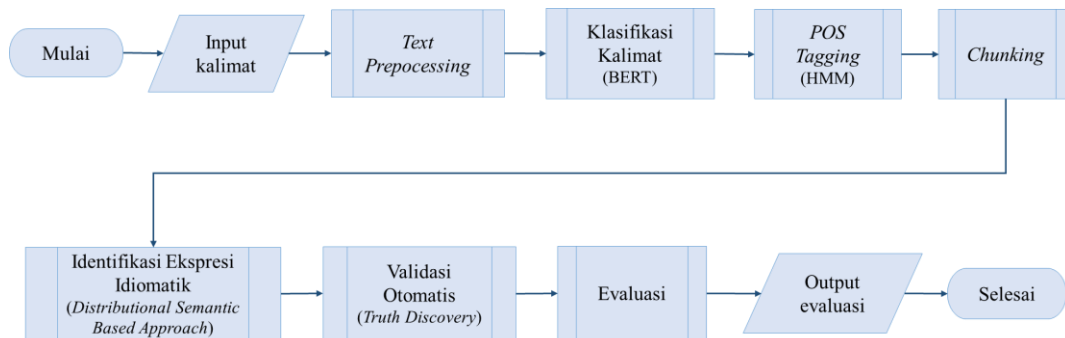
Data berikutnya yaitu data sumber web berupa situs-situs web yang membahas mengenai ekspresi idiomatik bahasa Indonesia yang berjumlah 104 halaman web dengan total jumlah klaim 4358 klaim. Data ini akan digunakan pada tahap validasi *Truth Discovery*. Contoh data ditunjukkan pada Tabel 3.5.

Tabel 3. 5 Contoh Data Situs Web

indeks	source_web	source_url	idiom_claims
1	salamadian.com	https://salamadian.com/contoh-ungkapan-bahasa-indonesia-dan-artinya-idiom/	Adu Mulut
2	salamadian.com	https://salamadian.com/contoh-ungkapan-bahasa-indonesia-dan-artinya-idiom/	Akal Bulus
3	salamadian.com	https://salamadian.com/contoh-ungkapan-bahasa-indonesia-dan-artinya-idiom/	Angkat Bicara
4	salamadian.com	https://salamadian.com/contoh-ungkapan-bahasa-indonesia-dan-artinya-idiom/	Angkat Kaki

3.3 Desain Sistem

Pada bagian ini, akan digambarkan alur secara umum dari penelitian yang akan dilakukan penulis, yaitu data berupa kumpulan kalimat berpola dasar berbahasa Indonesia yang tidak ataupun mengandung ekspresi idiomatik yang telah dikumpulkan dimasukkan ke dalam sistem, setelah itu akan dilakukan *text preprocessing*. Kemudian masuk ke tahap klasifikasi kalimat menggunakan BERT. Selanjutnya kalimat yang diklasifikasikan sebagai ‘kalimat_idiom’ akan masuk ke tahap *POS Tagging* dan *chunking*. Setelah mendapatkan hasil *chunking* berupa frasa, selanjutnya akan masuk ke tahap identifikasi ekspresi idiomatik menggunakan *Distributional Semantic Based Approach*. Setelah itu frasa yang telah diidentifikasi sebagai ekspresi idiomatik akan divalidasi secara otomatis menggunakan metode *Truth Discovery* yaitu kombinasi algoritma *Sums* dan *Average-Log*. Hasil identifikasi yang sudah divalidasi secara otomatis tersebut akan menjadi output akhir dari penelitian ini yang selanjutnya akan masuk ke tahap evaluasi. Secara umum, alur penelitian dapat dilihat pada Gambar 3.4.



Gambar 3. 4 Alur Umum Penelitian

3.3.1 Text Preprocessing

Pada penelitian ini, data kumpulan kalimat bahasa Indonesia akan dilakukan *text preprocessing* terlebih dahulu agar data tersebut siap dan dapat diolah untuk tahap selanjutnya. *Text preprocessing* digunakan untuk menyajikan data berupa teks dalam format yang sesuai. Adapun langkah-langkah yang dilakukan untuk *text preprocessing* pada penelitian ini adalah *punctuation removal*, *tokenization*, dan *case conversion*. Berikut akan dijelaskan masing-masing dari tahapan *text preprocessing* yang akan dilakukan.

a. *Punctuation removal*

Punctuation removal merupakan proses untuk menghilangkan simbol-simbol yang terdapat pada kalimat masukan. Simbol yang akan dihapus adalah !()-[]{};:'"\<>/?@#\$\$%^&*~. Sedangkan tanda baca titik dan koma tidak dihapus karena pada tahap *POS Tagging* simbol tersebut akan diberikan tag Z. Contoh penerapan *punctuation removal* dapat dilihat pada Tabel 3.6.

Tabel 3. 6 Contoh *Punctuation Removal*

Masukan	Hasil <i>Punctuation Removal</i>
Kutu buku itu membawa dua piala (emas dan perak) pada olimpiade nasional.	Kutu buku itu membawa dua piala emas dan perak pada olimpiade nasional.

b. *Tokenization*

Tokenization merupakan proses untuk memecah kalimat menjadi *token* yang dalam hal ini berupa kata. Contoh penerapan *tokenization* dapat dilihat pada Tabel 3.7.

Tabel 3. 7 Contoh *Tokenization*

Masukan	Hasil <i>Tokenization</i>	
Kutu buku itu membawa dua piala emas dan perak pada olimpiade nasional.	Kutu buku itu membawa dua piala	emas dan perak pada olimpiade nasional .

c. *Case conversion*

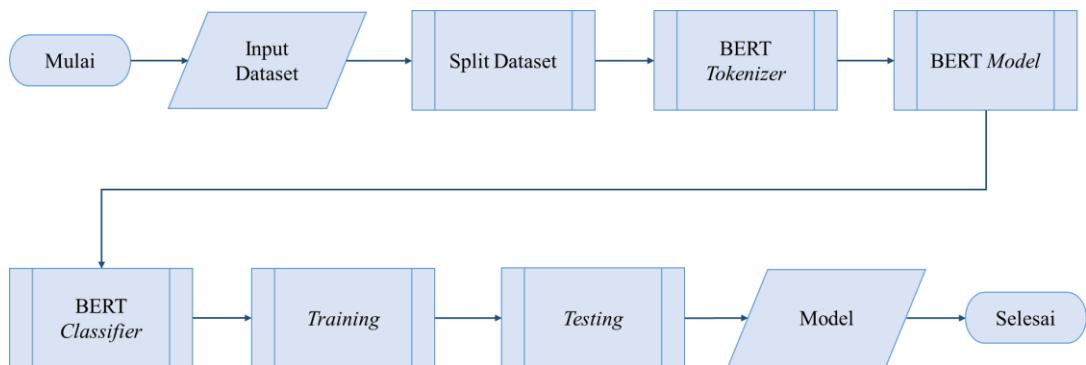
Case conversion adalah proses mengkonversi bentuk huruf dalam teks menjadi seragam baik itu menjadi huruf kecil atau huruf besar. Pada penelitian ini, frasa hasil identifikasi akan diseragamkan bentuknya menjadi huruf kecil (*lowercase*). Contoh penerapan *case conversion* dapat dilihat pada Tabel 3.8.

Tabel 3. 8 Contoh *Case Conversion*

Masukan	Hasil <i>Case Conversion</i>
Kutu buku	kutu buku

3.3.2 Klasifikasi Kalimat Menggunakan BERT

Pada tahap ini dilakukan klasifikasi terhadap kalimat inputan ke dalam dua kategori yaitu ‘kalimat_biasa’ jika kalimat tidak mengandung ekspresi idiomatik dan ‘kalimat_idiom’ jika kalimat mengandung ekspresi idiomatik. Tahap ini membutuhkan model klasifikasi yang dibangun menggunakan BERT. Gambar 3.5 menunjukkan alur proses pembuatan model klasifikasi berbasis BERT.



Gambar 3. 5 Alur Proses Pembuatan Model Klasifikasi

a. BERT Tokenizer

Pada tahap BERT *tokenizer*, setiap kalimat ditokenisasi menjadi per kata atau sub kata menggunakan *WordPiece*. Selanjutnya setiap kalimat inputan akan diberikan token-token khusus yaitu [CLS] di awal kalimat, token [SEP] di akhir kalimat dan token [PAD]. Output dari BERT *tokenizer* terdiri dari token berupa kata, *input_ids*, *attention_mask* dan *token_type_ids* yang akan menjadi inputan pada BERT *Model*. Contoh penerapan BERT *tokenizer* dengan *max_length* = 10 dapat dilihat pada tabel 3.9.

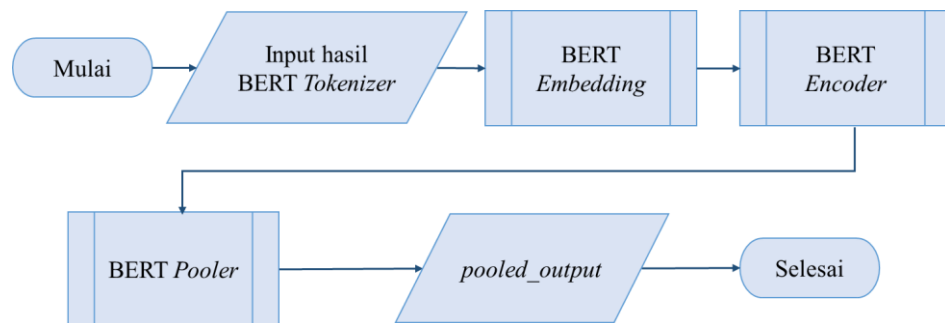
Tabel 3. 9 Contoh BERT tokenizer

kalimat	Andi dikenal sebagai kutu buku.
token	['[CLS]', 'andi', 'dikenal', 'sebagai', 'kutu', 'buku', '.', '[SEP]', '[PAD]', '[PAD]']
input_ids	[3, 10948, 2106, 1600, 6814, 2630, 17, 1, 2, 2]
attention_mask	[1, 1, 1, 1, 1, 1, 1, 1, 0, 0]
token_type_ids	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

b. BERT Model

Pada tahap ini, lapisan BERT *Model* dibuat berdasarkan arsitektur BERT yang ditunjukkan pada gambar Gambar 2. 2. BERT *Model* terdiri dari tiga lapisan yaitu BERT *Embedding* untuk merepresentasikan kalimat inputan

ke dalam bentuk vektor, BERT *Encoder* untuk menemukan *contextual information* dari tiap kata dalam inputan dan BERT *Pooler* yang akan mengambil output token [CLS] yang merupakan token klasifikasi khusus yang digunakan sebagai representasi untuk tugas klasifikasi yang kemudian diproses untuk menghasilkan *pooled_output*. Alur proses pada lapisan BERT *Model* dapat dilihat pada Gambar 3.6 berikut:



Gambar 3. 6 Alur Proses Lapisan BERT *Model*

c. BERT *Classifier*

Pada tahap ini, lapisan BERT *Classifier* dibuat berdasarkan pada gambar Gambar 2. 8. Pada lapisan BERT *Classifier* ini memberikan output hasil klasifikasi menggunakan fungsi *softmax*. Output yang dihasilkan akan bernilai 0 untuk kategori ‘kalimat_biasa’ dan 1 untuk kategori ‘kalimat_idiom’.

3.3.3 POS *Tagging*

Pada penelitian ini, *Part-of-Speech* (POS) *Tagging* digunakan untuk memberikan label pada setiap kata dalam kalimat dengan kelas kata yang sesuai untuk kata tersebut, seperti kata benda, kata kerja, kata sifat, dan lain-lain berdasarkan pada Tabel 2.2. Pada tahap *Part-of-Speech Tagging* ini membutuhkan model yang dibangun menggunakan algoritma *Hidden Markov Model* (HMM). *Part-of-Speech Tagging* pada penelitian ini dilakukan untuk memudahkan tahap selanjutnya yaitu *chunking* dimana kata-kata akan dikelompokkan ke dalam bentuk frasa yang ditentukan berdasarkan label kelas kata dari kata-kata tersebut. Contoh penerapan metode *Part-of-Speech Tagging* dapat dilihat pada Tabel 3.10 berikut.

Tabel 3. 10 Contoh *Part-of-Speech Tagging*

Masukan	Hasil POS Tagging
['Andi', 'dikenal', 'sebagai', 'kutu', 'buku', '.']	[('Andi', 'NNP'), ('dikenal', 'VB'), ('sebagai', 'IN'), ('kutu', 'NN'), ('buku', 'NN'), ('.', 'Z')]

3.3.4 *Chunking*

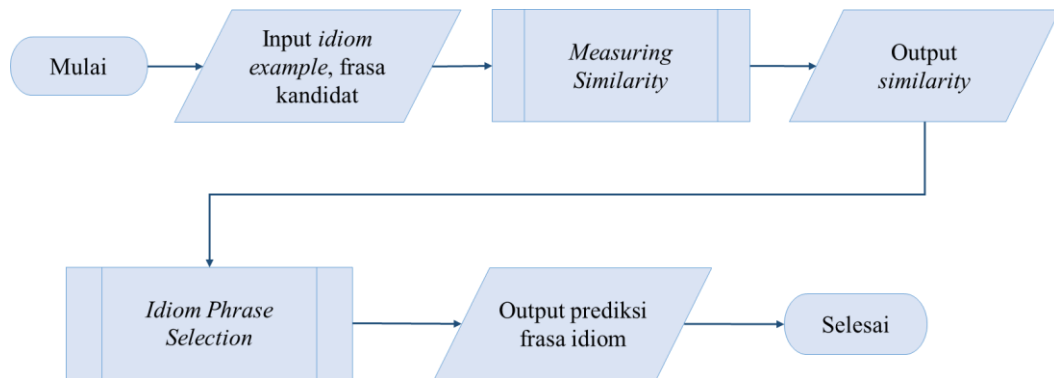
Pada penelitian ini, metode *chunking* digunakan untuk menemukan frasa nomina, frasa verba ataupun frasa adjektiva yang akan dikategorikan sebagai frasa kandidat dimana frasa tersebut memiliki konstruksi sintaksis pembentuk ekspresi idiomatik yang dapat dilihat pada Tabel 2.1. Untuk menemukan frasa kandidat tersebut pada suatu kalimat, penulis akan mendefinisikan tata bahasa *chunk* (*chunk grammar*) menggunakan *tag* dari *part-of-speech tagging*, yang terdiri dari aturan *Regular Expressions* yang menunjukkan bagaimana kalimat harus dipotong. Contoh penerapan metode *chunking* dapat dilihat pada Tabel 3.11.

Tabel 3. 11 Contoh *Chunking*

Masukan	Hasil <i>Chunking</i>
[('Andi', 'NNP'), ('dikenal', 'VB'), ('sebagai', 'IN'), ('kutu', 'NN'), ('buku', 'NN'), ('.', 'Z')]	kutu buku

3.3.5 Identifikasi Ekspresi Idiomatik Menggunakan *Distributional Semantic Based Approach*

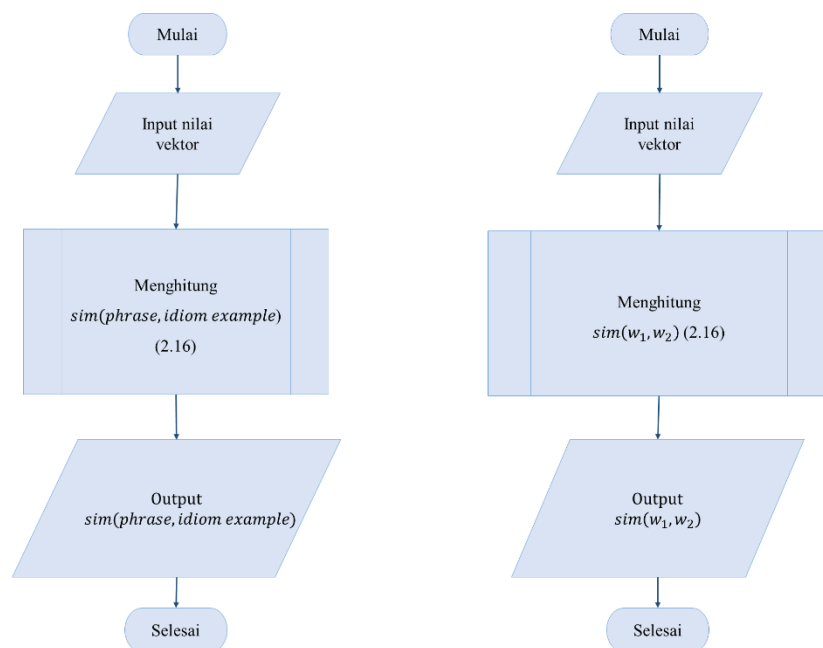
Pada tahap ini akan dilakukan identifikasi terhadap frasa-frasa kandidat yang dihasilkan dari proses *chunking* apakah frasa tersebut merupakan ekspresi idiomatik atau tidak menggunakan pendekatan berbasis semantik distribusi (*Distributional Semantic Based Approach*) dengan menjumlahkan kemiripan semantik antara kandidat dan kumpulan contoh ekspresi idiomatik dengan kemiripan semantik antara kata penyusun frasa kandidat berdasarkan pada persamaan (2.15).



Gambar 3. 7 Alur Proses Identifikasi Ekspresi Idiomatik

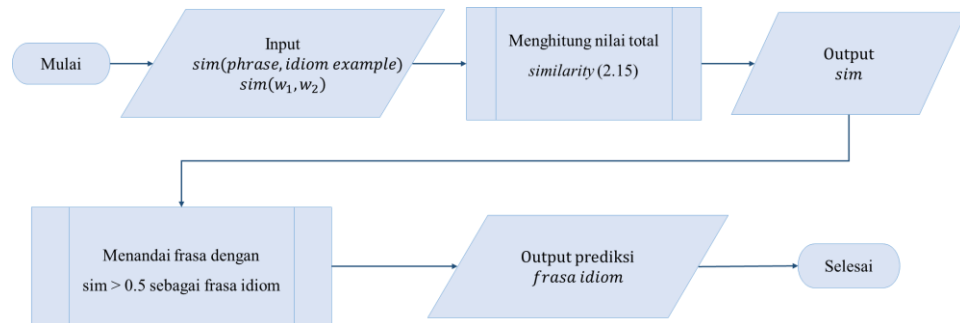
a. *Measuring Similarity*

Pada tahap ini akan dilakukan perhitungan nilai *similarity* dengan menggunakan *cosine similarity* berdasarkan persamaan (2.16). Perhitungan nilai *similarity* ini menerima inputan berupa representasi vektor dari kata-kata dan juga frasa yang didapatkan dari BERT *Embedding*. Perhitungan nilai *similarity* ini dibagi dalam dua bagian, yaitu *similarity* antar frasa kandidat (frasa hasil *chunking*) dengan contoh ekspresi idiomatik dan *similarity* antar kata penyusun frasa kandidat.

a. *Similarity* frasa kandidat dengan contoh idiomb. *Similarity* antar kata penyusun frasa kandidatGambar 3. 8 Alur Proses *Measuring Similarity*

b. *Idiom Phrase Selection*

Pada tahap ini, setelah mendapatkan hasil nilai *similarity* antar frasa kandidat dengan contoh idiom dan nilai *similarity* antar kata penyusun frasa kandidat, selanjutnya kedua nilai *similarity* tersebut dijumlahkan berdasarkan persamaan (2.15), kemudian akan dipilih dari *frasa-frasa* kandidat tersebut yang diidentifikasi sebagai frasa idiom berdasarkan frasa yang memiliki nilai *similarity* lebih besar dari batas yaitu 0,5.



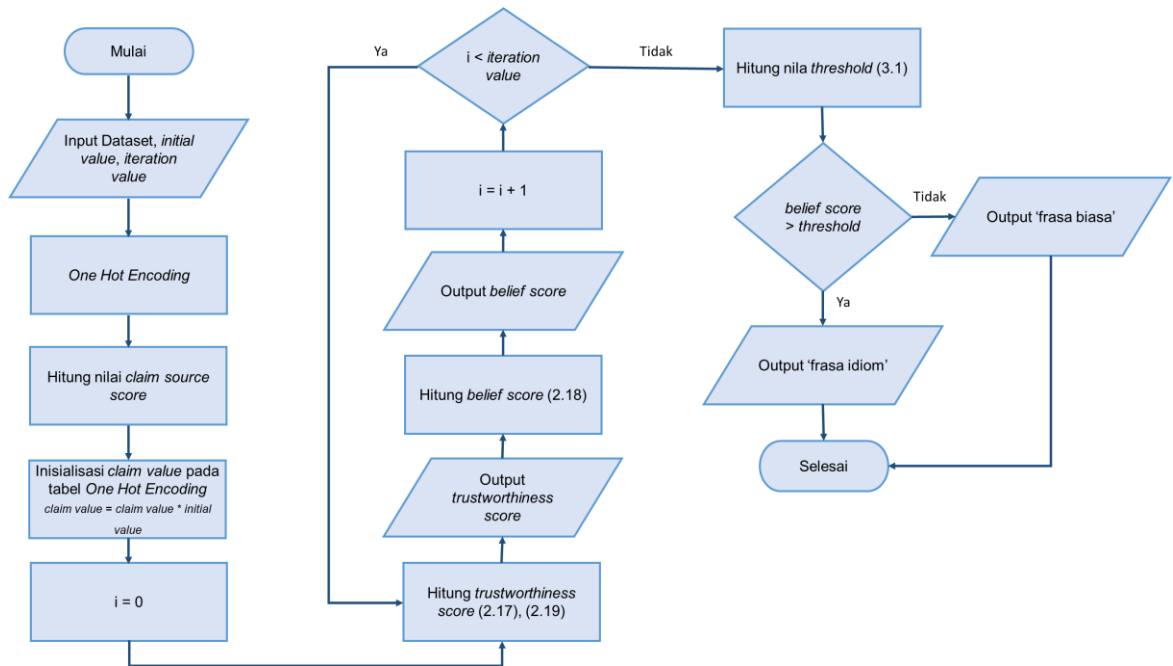
Gambar 3. 9 Alur Proses *Idiom Phrase Selection*

3.3.6 Validasi Ekspresi Idiomatik Menggunakan *Truth Discovery*

Pada tahap ini, frasa yang sudah diidentifikasi sebagai frasa ekspresi idiomatik akan divalidasi kebenarannya secara otomatis menggunakan metode *Truth Discovery* yaitu kombinasi algoritma *Sums* dan *Avarage-Log*. Algoritma *Sums* dan *Avarage-Log* akan menggunakan *trustworthiness score* dari sumber Website yang dipilih yang dihitung menggunakan kombinasi persamaan (2.17) dan (2.19) dan *belief score* dari setiap klaim yang diberikan oleh sumber yang dihitung menggunakan persamaan (2.18). Kemudian untuk memvalidasi suatu frasa merupakan frasa idiom akan menggunakan *belief score*, dimana suatu frasa dianggap benar atau valid sebagai frasa idiom jika mempunyai *belief score* di atas nilai parameter *threshold* (t). Nilai ambang batas atau *threshold* yang digunakan adalah berbasis persentil (Bornmann et al., 2012). Pada penelitian ini, nilai *threshold* didasarkan pada persentil 15 dari data *belief score* yang ditunjukkan oleh persamaan berikut:

$$t = \text{percentile}(\text{belief score}, 15) \quad (3.1)$$

Langkah-langkah melakukan validasi otomatis menggunakan *Truth Discovery* ditunjukkan oleh Gambar 3.10 berikut:



Gambar 3. 10 Alur Proses *Truth Discovery*

Langkah pertama yang dilakukan yaitu menginput data sumber web sebagai dataset, kemudian menginputkan *initial value* yang ditetapkan pada penelitian ini adalah 0.5 dan *iteration value* adalah 3. Kemudian membuat tabel *One Hot Encoding* yang berisikan *claim value* c pada sumber s , dimana *claim value* c akan bernilai 1 jika klaim c tersedia pada sumber s dan bernilai 0 untuk sebaliknya. Menghitung nilai *claim source score* C_s yaitu banyaknya jumlah klaim yang diberikan oleh setiap sumber s . Melakukan inisialisasi *claim value* = *claim value* \times *initial value*. Selanjutnya menghitung *trustworthiness score* sumber s dan *belief score* klaim c . Kemudian melakukan proses validasi frasa ekspresi idiomatik dengan membandingkan nilai *belief score* dengan *threshold*.

3.4 Desain Evaluasi Sistem

3.4.1 Skenario Pengujian

Pada penelitian ini akan dilakukan pengujian klasifikasi menggunakan BERT dan pengujian identifikasi menggunakan *Truth Discovery*. Berikut ini merupakan skenario pengujian yang dilakukan yaitu:

a. Pengujian klasifikasi menggunakan BERT

Tahapan pengujian ini akan dilakukan dengan menggunakan metode *K-Fold Cross-Validation*, dengan $k = 5$. Seluruh data kalimat Bahasa Indonesia yang ada akan dibagi menjadi 5 bagian atau *folds*. Pelatihan akan dilakukan secara berulang sebanyak 5 kali, setiap pengulangan yang dilakukan, 4 bagian akan dijadikan data pembelajaran dan satu bagian akan dijadikan data uji. Proses pengujian klasifikasi BERT dilakukan berdasarkan konfigurasi *hyperparameter* yang dapat dilihat pada Tabel 3.12. Pengujian ini digunakan untuk mendapatkan model terbaik.

Tabel 3. 12 Konfigurasi *Hyperparameter*

<i>Hyperparameter</i>	Nilai Tetapan
Dropout	0,5
Learning Rate	2e-4, 3e-4, 2e-5, 3e-5, 2e-6, 3e-6
Batch Size	16
Epoch	3, 4, 5, 6, 7

b. Pengujian identifikasi menggunakan *Truth Discovery*

Tahapan pengujian ini akan dilakukan 10 kali pengujian. Pada penelitian ini, seluruh data kalimat Bahasa Indonesia digunakan sebagai data uji, hal ini terjadi karena dalam identifikasi yang tidak memerlukan data latih. Pada setiap pengujian memiliki jumlah data uji yang berbeda-beda. Selanjutnya hasil evaluasi didapatkan melalui rata-rata dari 10 kali pengujian. Hasil dari pengujian akan dituangkan pada tabel berikut.

Tabel 3. 13 Contoh Tabel Hasil Pengujian Identifikasi

Pengujian	Jumlah Data	Akurasi	Presisi	Recall	F1-Score
1	200	p	q	r	s
2	400	p	q	r	s
3	600	p	q	r	s
4	800	p	q	r	s
5	1000	p	q	r	s
6	1200	p	q	r	s
7	1400	p	q	r	s
8	1600	p	q	r	s
9	1800	p	q	r	s
10	2000	p	q	r	s
Rata-rata		p	q	r	s

3.4.2 Evaluasi Sistem

Untuk mengetahui performa model terbaik yang didapatkan, diperlukan sebuah teknik untuk pengukuran evaluasi. Pada penelitian ini akan dilakukan pengukuran performa menggunakan *confusion matrix*. Parameter yang digunakan adalah *accuracy*, *precision*, *recall* dan *F1-score*. Perhitungan *accuracy* menggunakan persamaan (2.20), perhitungan *precision* menggunakan persamaan (2.21), perhitungan *recall* menggunakan persamaan (2.22) dan perhitungan *F1-score* menggunakan persamaan (2.23).

BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi Sistem

4.1.1 Implementasi *Text Preprocessing*

Pada penelitian ini, data kumpulan kalimat bahasa Indonesia akan dilakukan *text preprocessing* terlebih dahulu agar data tersebut siap dan dapat diolah untuk tahap selanjutnya. *Text preprocessing* digunakan untuk menyajikan data berupa teks dalam format yang sesuai. Tahapan *text preprocessing* pada penelitian ini terdiri dari *punctuation removal*, *tokenization*, dan *case conversion*. Penggalan kode proses *text preprocessing* dapat dilihat pada Tabel 4.1. Tahap *punctuation removal* dilakukan dengan menggunakan fungsi `maketrans` dan `translate` yang ditunjukkan pada baris ke-3 sampai baris ke-5, tahapan *tokenization* dilakukan dengan menggunakan `WordPunctTokenizer()` dari library `nltk` yang ditunjukkan pada baris ke-7 sampai baris ke-8, dan tahapan *case conversion* menggunakan fungsi `lower` yang ditunjukkan pada baris 10.

Tabel 4. 1 Penggalan Kode Proses *Text Preprocessing*

No	Penggalan Kode
1	<code>def text_preprocessing(self, kalimat, r =False, t=False, l=False):</code>
2	<code> if (r):</code>
3	<code> punc = ' '!()-[]{};:'"<>/?@#\$\$%^&* ~'''</code>
4	<code> kalimat = kalimat.translate(str.maketrans(' ', '', punc))</code>
5	<code> kalimat = re.sub(r'/s+', ' ', kalimat).strip()</code>
6	<code> if (t):</code>
7	<code> word_punct_tokenizer = WordPunctTokenizer()</code>
8	<code> kalimat = word_punct_tokenizer.tokenize(kalimat)</code>
9	<code> if (l):</code>
10	<code> kalimat = str.lower(kalimat)</code>
11	<code> return kalimat</code>

4.1.2 Implementasi Klasifikasi Kalimat Menggunakan BERT

Tahapan selanjutnya yaitu klasifikasi kalimat ke dalam dua kategori yaitu 'kalimat_biasa' jika kalimat tidak mengandung ekspresi idiomatik dan 'kalimat_idiom' jika kalimat mengandung ekspresi idiomatik. Penggalan kode proses klasifikasi kalimat dapat dilihat pada Tabel 4.2. Pada tahapan ini, pertama-tama kalimat masukan akan melalui `tokenizer` dari BERT untuk menghasilkan

`input_ids`, `attention_mask`, dan `token_type_ids` sebagai inputan pada model ditunjukkan pada baris ke-2 sampai baris ke-15. Setelah itu proses klasifikasi dilakukan dengan memanggil `classification_model` yang telah dibuat sebelumnya menggunakan BERT yang dapat dilihat pada baris ke-18, kemudian hasil prediksi dari model akan disimpan pada variabel `kategori`.

Tabel 4. 2 Penggalan Kode Proses Klasifikasi

No	Penggalan Kode
1	<code>def idiom_sentence_classification(self, kalimat):</code>
2	<code> encoded_text = self.tokenizer.encode_plus(</code>
3	<code> kalimat,</code>
4	<code> max_length=40,</code>
5	<code> add_special_tokens=True,</code>
6	<code> return_token_type_ids=True,</code>
7	<code> padding='max_length',</code>
8	<code> truncation=True,</code>
9	<code> return_attention_mask=True,</code>
10	<code> return_tensors='pt',</code>
11	<code>)</code>
12	
13	<code> input_ids = encoded_text['input_ids'].to(self.device)</code>
14	<code> attention_mask = encoded_text['attention_mask'].to(self.device)</code>
15	<code> token_type_ids = encoded_text['token_type_ids'].to(self.device)</code>
16	
17	<code> output, pred</code>
18	<code>= classification_model(input_ids, attention_mask, token_type_ids)</code>
19	<code> kategori = self.class_names[prediction]</code>
20	
21	<code> return kategori</code>

Selanjutnya untuk proses pembuatan model klasifikasi menggunakan BERT dilakukan berdasarkan *flowchart* pada Gambar 3.5 yang terdiri dari beberapa lapisan yaitu BERT *Tokenizer*, BERT *Model* dan BERT *Classifier*. Pada lapisan BERT *Tokenizer* yang penggalan kodenya ditunjukkan pada Tabel 4.3, pada baris ke-3 sampai baris ke-7 menerima inputan berupa `texts` yaitu kalimat berbahasa Indonesia, `labels` yaitu label dari kalimat, `tokenizer` yaitu fungsi yang digunakan untuk melakukan tokenisasi dalam hal ini yaitu `BertTokenizer` dari library `transformers` dan `max_len` yaitu maksimal panjang dari inputan sekuen dimana dalam penelitian ini adalah 40. Selanjutnya untuk proses tokenisasi dilakukan pada baris ke-16 sampai baris ke-25. Kemudian pada lapisan BERT *Tokenizer* ini menghasilkan `output` yang terdiri dari `text`, `input_ids`, `attention_mask`, `token_type_ids` dan `labels` yang akan menjadi inputan pada lapisan BERT *Model*.

Tabel 4. 3 Penggalan Kode Lapisan *Bert Tokenizer*

No	Penggalan Kode
1	<code>class BertTokenize(Dataset):</code>
2	
3	<code>def __init__(self, texts, labels, tokenizer, max_len):</code>
4	<code>self.texts = texts</code>
5	<code>self.labels = labels</code>
6	<code>self.tokenizer = tokenizer</code>
7	<code>self.max_len = max_len</code>
8	
9	<code>def __len__(self):</code>
10	<code>return len(self.texts)</code>
11	
12	<code>def __getitem__(self, item):</code>
13	<code>text = str(self.texts[item])</code>
14	<code>label = self.labels[item]</code>
15	
16	<code>encoding = self.tokenizer.encode_plus(</code>
17	<code>text,</code>
18	<code>add_special_tokens=True,</code>
19	<code>max_length=self.max_len,</code>
20	<code>return_token_type_ids=True,</code>
21	<code>padding='max_length',</code>
22	<code>truncation=True,</code>
23	<code>return_attention_mask=True,</code>
24	<code>return_tensors='pt',</code>
25	<code>)</code>
26	
27	<code>return {</code>
28	<code>'text': text,</code>
29	<code>'input_ids': encoding['input_ids'].flatten(),</code>
30	<code>'token_type_ids': encoding['token_type_ids'].flatten(),</code>
31	<code>'attention_mask': encoding['attention_mask'].flatten(),</code>
32	<code>'labels': torch.tensor(label, dtype=torch.long)</code>
33	<code>}</code>

Selanjutnya adalah lapisan *BERT Model* yang penggalan kodenya ditunjukkan pada Tabel 4.4. Berdasarkan Gambar 2. 2, *BERT Model* terdiri dari tiga lapisan yaitu *embedding*, *encoder* dan *pooler*, ketiga lapisan tersebut diinisialisasikan dengan memanggil fungsi `BertEmbeddings`, `BertEncoder` dan `BertPooler` dari library `transformers` terlihat pada baris ke-2 sampai baris ke-8. Pada proses *embedding* menerima inputan dari hasil proses tokenisasi kemudian merepresentasikannya ke dalam bentuk vektor yang ditunjukkan pada baris ke-31 sampai baris ke-37. Selanjutnya output dari *embedding* akan diproses dalam lapisan *encoder* yang ditunjukkan pada baris ke-38 sampai baris ke-39. Kemudian output *encoder* akan masuk ke lapisan *pooler* yang ditunjukkan pada baris ke-50 sampai baris ke-51. Output dari lapisan *pooler* ini akan disimpan pada variabel `pooled_output` yang kemudian menjadi output dari lapisan *BERT Model*.

Tabel 4. 4 Penggalan Kode Lapisan *Bert Model*

No	Penggalan Kode
1	<code>class BertModel(BertPreTrainedModel):</code>
2	<code>def __init__(self, config):</code>
3	<code>super().__init__(config)</code>
4	<code>self.config = config</code>
5	<code>self.embeddings = BertEmbeddings(config)</code>
6	<code>self.encoder = BertEncoder(config)</code>
7	<code>self.pooler = BertPooler(config)</code>
8	<code>self.init_weights()</code>
9	
10	<code>def forward(self, input_ids=None, attention_mask=None,</code>
11	<code>token_type_ids=None, position_ids=None, head_mask=None,</code>
12	<code>inputs_embeds=None, encoder_hidden_states=None,</code>
13	<code>encoder_attention_mask=None, past_key_values=None,</code>
14	<code>use_cache=None, output_attentions=None,</code>
15	<code>output_hidden_states=None, return_dict=None,</code>
16	<code>):</code>
17	<code>output_attentions = self.config.output_attentions</code>
18	<code>output_hidden_states = self.config.output_hidden_states</code>
19	<code>return_dict = return_dict</code>
20	<code>use_cache = False</code>
21	
22	<code>if input_ids is not None:</code>
23	<code>input_shape = input_ids.size()</code>
24	<code>batch_size, seq_length = input_shape</code>
25	<code>device = input_ids.device</code>
26	<code>past_key_values_length = 0</code>
27	<code>extended_attention_mask: torch.Tensor =</code>
28	<code>self.get_extended_attention_mask(attention_mask, input_shape,</code>
29	<code>device)</code>
30	<code>encoder_extended_attention_mask = encoder_attention_mask</code>
31	<code>head_mask = self.get_head_mask(head_mask,</code>
32	<code>self.config.num_hidden_layers)</code>
33	<code>embedding_output = self.embeddings(</code>
34	<code>input_ids=input_ids,</code>
35	<code>position_ids=position_ids,</code>
36	<code>token_type_ids=token_type_ids,</code>
37	<code>inputs_embeds=inputs_embeds,</code>
38	<code>past_key_values_length=past_key_values_length,</code>
39	<code>)</code>
40	<code>encoder_outputs = self.encoder(</code>
41	<code>embedding_output,</code>
42	<code>attention_mask=extended_attention_mask,</code>
43	<code>head_mask=head_mask,</code>
44	<code>encoder_hidden_states=encoder_hidden_states,</code>
45	<code>encoder_attention_mask=encoder_extended_attention_mask,</code>
46	<code>past_key_values=past_key_values,</code>
47	<code>use_cache=use_cache,</code>
48	<code>output_attentions=output_attentions,</code>
49	<code>output_hidden_states=output_hidden_states,</code>
50	<code>return_dict=return_dict,</code>
51	<code>)</code>
52	<code>sequence_output = encoder_outputs[0]</code>
53	<code>pooled_output = self.pooler(sequence_output)</code>
54	<code>if not return_dict:</code>
	<code>return (sequence_output, pooled_output) + encoder_outputs[1:]</code>

Lapisan berikutnya adalah lapisan *BERT Classifier* yang penggelan kodenya dapat dilihat pada Tabel 4.5. Berdasarkan Gambar 2.8, *BERT Classifier* terdiri dari tiga lapisan yaitu lapisan *Bert Model* yang telah dibuat sebelumnya, serta lapisan *Drouput* dan *Linear* dengan menggunakan `nn.Dropout` dan `nn.Linear` dari library `PyTorch` yang dapat dilihat pada baris ke-3 sampai baris ke-7. Selanjutnya pada baris ke-10 sampai baris ke-15 lapisan *BERT Model* dipanggil untuk memproses inputan dan menghasilkan output yang disimpan dalam variabel `pooled_output`. Kemudian `pooled_output` akan masuk ke dalam lapisan *Dropout* dan *Linear* sehingga menghasilkan output yaitu `logits` berupa prediksi probabilitas kasar dari kalimat yang akan diklasifikasikan, proses ini ditunjukkan pada baris ke-16 sampai baris ke-17. Selanjutnya proses klasifikasi dilakukan dengan menggunakan fungsi `softmax` dengan `logits` sebagai inputan. Proses klasifikasi ini menghasilkan hasil prediksi label dari kalimat inputan yang disimpan pada variabel `pred`. Variabel ini akan bernilai 0 sebagai ‘kalimat_biasa’ atau 1 sebagai ‘kalimat_idiom’.

Tabel 4. 5 Penggalan Kode Lapisan *Bert Classifier*

No	Penggalan Kode
1	<code>class BertClassifier(nn.Module):</code>
2	
3	<code>def __init__(self, n_classes, dropout=0.3):</code>
4	<code>super(BertClassifier, self).__init__()</code>
5	<code>self.bert = BertModel.from_pretrained(MODEL_NAME)</code>
6	<code>self.drop = nn.Dropout(p=dropout)</code>
7	<code>self.out = nn.Linear(self.bert.config.hidden_size, n_classes)</code>
8	
9	<code>def forward(self, input_ids, attention_mask, token_type_ids):</code>
10	<code>_, pooled_output = self.bert(</code>
11	<code>input_ids=input_ids,</code>
12	<code>attention_mask=attention_mask,</code>
13	<code>token_type_ids=token_type_ids,</code>
14	<code>return_dict=False</code>
15	<code>)</code>
16	<code>output = self.drop(pooled_output)</code>
17	<code>logits = self.out(output)</code>
18	<code>classifier = torch.nn.functional.softmax(logits, dim=1)</code>
19	<code>_, pred = torch.max(classifier, dim=1)</code>
20	<code>return logits, pred</code>

Setelah selesai membuat lapisan *BERT Tokenizer*, *BERT Model* dan *BERT Classifier*. Proses selanjutnya dalam pembuatan model klasifikasi yaitu proses *training* dan *testing*. Proses ini menggunakan dataset kalimat Bahasa Indonesia yang dapat dilihat pada Tabel 3.4, dataset ini akan dibagi dengan perbandingan 80%

untuk data *training* dan 20% data *testing*. Setelah mendapatkan performa yang baik selanjutnya model akan disimpan dan digunakan pada tahap klasifikasi kalimat.

4.1.3 Implementasi POS Tagging

Setelah melalui tahapan klasifikasi menggunakan BERT, akan dilanjutkan ke tahapan POS Tagging. Pada penelitian ini, POS Tagging digunakan untuk memberikan label pada setiap kata dalam kalimat dengan kelas kata berdasarkan pada Tabel 2.2. Penggalan kode proses POS Tagging dapat dilihat pada Tabel 4.6. Proses melabeli kata dengan kelas katanya diawali dengan memanggil model `hmm_tagger` yang telah dibuat sebelumnya yang ditunjukkan pada baris ke-3. Model ini akan menerima inputan berupa hasil tokenisasi dari kalimat. Selanjutnya hasil prediksi kelas kata yang dihasilkan model ini akan disimpan ke dalam variabel `tagging`. Setelah itu prediksi kelas kata pada variabel `tagging` akan di periksa kembali menggunakan fungsi `check_tag` dan disimpan dalam variabel `final_tag` yang ditunjukkan pada baris ke-5 sampai baris ke-7.

Tabel 4. 6 Penggalan Kode Proses POS Tagging

No	Penggalan Kode
1	<code>def pos_tagging(self, kalimat):</code>
2	<code> kalimat_token=self.text_preprocessing(kalimat, t=True)</code>
3	<code> tagging = self.hmm_tagger_model.tag(kalimat_token)</code>
4	<code> final_tag = []</code>
5	<code> for pt in tagging:</code>
6	<code> w,t = self.check_tag(pt[0], pt[1])</code>
7	<code> final_tag.append((w,t))</code>
8	
9	<code> return final_tag</code>

Seperti yang sudah dijelaskan sebelumnya, pada tahap *Part-of-Speech Tagging* ini membutuhkan model yang dibangun menggunakan algoritma *Hidden Markov Model* (HMM). Penggalan kode proses pembuatan model POS tagging ini dapat dilihat pada Tabel 4.7. Proses pembuatan model diawali dengan membuka dan membaca dataset `Indonesian_Manually_Tagged_Corpus.tsv` yaitu kumpulan kalimat bahasa Indonesia yang telah diberikan kelas kata secara manual yang dapat dilihat pada baris ke-1 sampai baris ke-3. Setelah itu kalimat-kalimat beserta kelas katanya yang terdapat pada dataset akan dimuat dalam variabel `sentences` yang ditunjukkan pada baris ke-5 sampai baris ke-13.

Kemudian data dalam variabel `sentences` akan dibagi dengan perbandingan 80% data *training* dan 20% data *testing* yang dapat dilihat pada baris ke-15 sampai baris ke-17. Selanjutnya proses *training* dan *testing* model dilakukan dengan memanggil fungsi `HiddenMarkovModelTrainer()` dari library `nlTK.tag.hmm` yang ditunjukkan pada baris ke-19 sampai baris ke-23. Setelah proses *training* dan *testing* selesai dan mendapatkan performa yang baik, selanjutnya model disimpan dan akan digunakan pada tahap POS *tagging*.

Tabel 4. 7 Penggalan Kode Proses Pembuatan Model POS *Tagging*

No	Penggalan Kode
1	<code>fname = "Indonesian_Manually_Tagged_Corpus.tsv"</code>
2	<code>fopen = open(fname, 'r')</code>
3	<code>fdata = fopen.readlines()</code>
4	
5	<code>sentences = []</code>
6	<code>sentence = []</code>
7	<code>for word in fdata:</code>
8	<code> dt = word.replace('\n', '').split('\t')</code>
9	<code> if(len(dt) == 1):</code>
10	<code> sentences.append(sentence)</code>
11	<code> sentence = []</code>
12	<code> else:</code>
13	<code> sentence.append((dt[0],dt[1]))</code>
14	
15	<code>cutoff = int(.8 * len(sentences))</code>
16	<code>training_sentences = sentences[:cutoff]</code>
17	<code>test_sentences = sentences[cutoff:]</code>
18	
19	<code>trainer = HiddenMarkovModelTrainer()</code>
20	<code>tagger = trainer.train_supervised(training_sentences,\</code>
21	<code> estimator=lambda fd,\</code>
22	<code> bins: LidstoneProbDist(fd, 0.1, bins))</code>
23	<code>tagger.test(test_sentences, verbose=True)</code>

4.1.4 Implementasi *Chunking*

Setelah melalui tahapan POS *Tagging*, akan dilanjutkan ke tahapan *Chunking*. Pada penelitian ini, metode *chunking* digunakan untuk menemukan frasa nomina, frasa verba ataupun frasa adjektiva yang akan dikategorikan sebagai frasa kandidat dimana frasa tersebut memiliki konstruksi sintaksis pembentuk ekspresi idiomatik. Penggalan kode proses *Chunking* dapat dilihat pada Tabel 4.8. Pada tahapan ini, diawali dengan membuat tata bahasa *chunk* (*chunk grammar*) yang terdiri dari aturan *Regular Expressions* yang ditunjukkan pada baris ke-2 sampai baris ke-6. *Chunk grammar* ini dibuat berdasarkan pada konstruksi sintaksis pembentuk ekspresi idiomatik yang dapat dilihat pada Tabel 2.1. Proses *chunking*

dilakukan dengan menggunakan library `nltk.RegexpParser` yang menerima inputan berupa hasil dari tahapan *POS Tagging*. Kemudian frasa-frasa yang didapatkan dari hasil *chunking* akan disimpan dalam variabel `frasa_kandidat`, proses ini dapat dilihat pada baris ke-8 samapi baris ke-23.

Tabel 4. 8 Penggalan Kode Proses *Chunking*

No	Penggalan Kode
1	<code>def chunking(self, kalimat_tagged):</code>
2	<code> grammar = ["CHUNK: {<NN>{2,}}", "CHUNK: {<NN><JJ>}",</code>
3	<code> "CHUNK: {<NN><VB>}", "CHUNK: {<CD><NN>}",</code>
4	<code> "CHUNK: {<VB><VB>}", "CHUNK: {<VB><NN>}",</code>
5	<code> "CHUNK: {<VB><JJ>}", "CHUNK: {<VB><CD>}",</code>
6	<code> "CHUNK: {<JJ><NN>}", "CHUNK: {<JJ><JJ>}"]</code>
7	
8	<code> frasa_kandidat = []</code>
9	
10	<code> for i in grammar:</code>
11	<code> cp = nltk.RegexpParser(i)</code>
12	<code> result = cp.parse(kalimat_tagged)</code>
13	
14	<code> leaves = [chunk.leaves() for chunk in result \</code>
15	<code> if ((type(chunk) == nltk.tree.Tree) and \</code>
16	<code> chunk.label() == 'CHUNK')]</code>
17	<code> bigram_groups = [list(nltk.bigrams([w for w, t in leaf])) \</code>
18	<code> for leaf in leaves]</code>
19	
20	<code> fr = [' '.join(w) for group in bigram_groups for w in group]</code>
21	<code> frasa_kandidat = frasa_kandidat + fr</code>
21	
23	<code> return frasa_kandidat</code>

4.1.5 Implementasi Identifikasi Ekspresi Idiomatik Menggunakan

Distributional Semantic Based Approach

Pada tahap ini akan dilakukan identifikasi terhadap frasa-frasa kandidat yang dihasilkan dari proses *chunking* apakah frasa tersebut merupakan ekspresi idiomatik atau tidak menggunakan pendekatan berbasis semantik distribusi (*Distributional Semantic Based Approach*) berdasarkan persamaan (2.15). Penggalan kode proses identifikasi ini dapat dilihat pada Tabel 4.9. Berdasarkan pada Gambar 3.6, proses identifikasi terdiri dari dua tahap yaitu *measuring similarity* dan *idiom phrase selection*.

Pada tahap *measuring similarity* akan dilakukan dengan memanggil `similarity_model` yang telah dibuat sebelumnya. Berdasarkan Gambar 3.7, perhitungan nilai *similarity* ini dibagi dalam dua bagian, yaitu *similarity* antar frasa kandidat dengan contoh ekspresi idiomatik yang ditunjukkan pada baris ke-8 sampai

baris ke-9 dan *similarity* antar kata penyusun frasa kandidat yang ditunjukkan pada baris ke-3.

Selanjutnya pada tahap *idiom phrase selection*, setelah mendapatkan hasil total nilai *similarity* frasa kandidat, kemudian frasa-frasa kandidat yang memiliki nilai *similarity* lebih besar dari batas yaitu 0,5 akan diidentifikasi sebagai ekspresi idiomatik dan disimpan dalam variabel *frasa_pred* yang ditunjukkan pada baris ke-19 sampai baris ke-22.

Tabel 4. 9 Penggalan Kode Proses *Distributional Semantic Based Approach*

No	Penggalan Kode
1	<code>def count_score_similarity(self, frasa):</code>
2	<code> token = frasa.split()</code>
3	<code> similarity_score=self.similarity_model.predict(token[0],token[1])</code>
4	<code> [0][0]</code>
5	<code> random = self.idiom_example_df.sample(n = 5)</code>
6	<code> idiom_example = random['idiom_example'].values</code>
7	<code> for idiom in idiom_example:</code>
8	<code> similarity_score = similarity_score + self.similarity_model.pre</code>
9	<code>dict(frasa, idiom)[0][0]</code>
10	<code> return similarity_score</code>
11	<code> return similarity_score</code>
12	<code> return similarity_score</code>
13	<code> return similarity_score</code>
14	<code>def similarity(self, frasa):</code>
15	<code> frasa_pred = []</code>
16	<code> if len(frasa) == 0:</code>
17	<code> frasa_pred = []</code>
18	<code> else:</code>
19	<code> for f in frasa:</code>
20	<code> sim_score = self.count_score_similarity(f)</code>
21	<code> if sim_score > 0.5:</code>
22	<code> frasa_pred.append(f)</code>
23	<code> frasa_pred.append(f)</code>
24	<code> return frasa_pred</code>

Seperti yang sudah dijelaskan sebelumnya, pada tahap perhitungan nilai *similarity* membutuhkan *similarity_model* yang dibuat berdasarkan representasi vektor dari kata-kata dan juga frasa yang dihasilkan dari BERT *Embedding*. Penggalan kode proses pembuatan model ini dapat dilihat pada Tabel 4.10. Proses *embedding* dapat dilihat pada baris ke-6 sampai baris ke-23. Proses perhitungan nilai *similarity* menggunakan *cosine similarity* berdasarkan persamaan (2.16) ditunjukkan pada baris ke-25 sampai baris ke-34.

Tabel 4. 10 Penggalan Kode Proses Pembuatan *Similarity Model*

No	Penggalan Kode
1	<code>class TokenSimilarity:</code>
2	<code>def __init__(self, from_pretrained:str=MODEL_NAME):</code>
3	<code>self.tokenizer = AutoTokenizer.from_pretrained(from_pretrained)</code>
4	<code>self.model = AutoModel.from_pretrained(from_pretrained)</code>
5	
6	<code>def __process(self, first_token, second_token):</code>
7	<code>inputs = self.tokenizer([first_token, second_token],</code>
8	<code>max_length=self.max_length,</code>
9	<code>truncation=self.truncation,</code>
10	<code>padding=self.padding,</code>
11	<code>return_tensors='pt')</code>
12	
13	<code>attention = inputs.attention_mask</code>
14	<code>outputs = self.model(**inputs)</code>
15	<code>embeddings = outputs[0]</code>
16	<code>mask = attention.unsqueeze(1).expand(embeddings.shape).float()</code>
17	<code>masked_embeddings = embeddings * mask</code>
18	
19	<code>summed = masked_embeddings.sum(1)</code>
20	<code>counts = clamp(mask.sum(1), min=1e-9)</code>
21	<code>mean_pooled = summed/counts</code>
22	
23	<code>return mean_pooled.detach().numpy()</code>
24	
25	<code>def predict(self, first_token, second_token, max_length=40,</code>
26	<code>truncation=True, padding="max_length"):</code>
27	<code>self.max_length = max_length</code>
28	<code>self.truncation = truncation</code>
29	<code>self.padding = padding</code>
30	
31	<code>mean_pooled=self.__process(first_token, second_token)</code>
32	<code>similarity=cosine_similarity([mean_pooled[0]], [mean_pooled[1]])</code>
33	
34	<code>return similarity</code>

4.1.6 Implementasi Validasi Ekspresi Idiomatik Menggunakan *Truth*

Discovery

Pada tahap ini, frasa yang sudah diidentifikasi sebagai frasa ekspresi idiomatik akan divalidasi kebenarannya secara otomatis menggunakan *Truth Discovery* yaitu kombinasi algoritma *Sums* dan *Average-Log*. Penggalan proses validasi menggunakan *Truth Discovery* dapat dilihat pada Tabel 4.11. Proses validasi dilakukan dengan memanggil `truth_discovery_model`, kemudian jika model mengembalikan nilai 1 maka frasa hasil identifikasi dianggap valid sebagai frasa idiom. Pada tahap ini memerlukan model validasi otomatis *Truth Discovery* yang dibuat berdasarkan langkah-langkah pada Gambar 3.9 dengan menggunakan data situs-situs web yang membahas mengenai ekspresi idiomatik, dan parameter *initial value* yang ditetapkan yaitu 0.5 dan *iteration value* adalah 3.

Tabel 4. 11 Penggalan Kode Proses Validasi *Truth Discovery*

No	Penggalan Kode
1	def validasi(self, frasa):
2	frasa_idiom = []
3	for f in frasa:
4	f = self.text_preprocessing(f, lowercase=True)
5	kategori = self.truth_discovery_model.predict([f])[0]
6	if kategori == 1:
7	frasa_idiom.append(f)
8	
9	return frasa_idiom

Langkah pertama yaitu membuat tabel *One Hot Encoding* yang berisikan *claim value* c pada sumber s , dimana *claim value* c akan bernilai 1 jika klaim c tersedia pada sumber s dan bernilai 0 untuk sebaliknya. Penggalan proses membuat tabel One Hot Encoding dapat dilihat pada Tabel 4.12.

Tabel 4. 12 Penggalan Kode Membuat Tabel *One Hot Encoding*

No	Penggalan Kode
1	def one_hot_encoding(self, source_list, claim_list, df):
2	scores = []
3	for sr in source_list:
4	s = []
5	for c in claim_list:
6	if c in data[data["source_url"] == sr].idiom_claims.values.tolist():
7	s.append(1)
8	else:
9	s.append(0)
10	scores.append(s)
11	claim = pd.DataFrame(scores, columns=claim_list)
12	source = pd.DataFrame([s] for s in source_list, columns=['source_url'])
13	data_score = pd.concat([source, claim], axis=1)
14	return scores, data_score

Langkah kedua yaitu menghitung nilai *claim source score* C_s yaitu banyaknya jumlah klaim yang diberikan oleh setiap sumber s . Penggalan kode proses menghitung *claim source score* dapat dilihat pada Tabel 4.13.

Tabel 4. 13 Penggalan Kode Menghitung *Claim Source Score* (C_s)

No	Penggalan Kode
1	def calculate_cs_score(self, scores):
2	Cs_count_list = []
3	for i in scores:
4	Cs = 0
5	for j in i:
6	if j == 1:
7	Cs = Cs + 1
8	Cs_count_list.append(Cs)
9	return Cs_count_list

Langkah selanjutnya yaitu melakukan inisialisasi *claim value* c pada setiap sumber s pada tabel *One Hot Encoding* dimana setiap *claim value* c akan dikalikan dengan *initial value* yang telah ditetapkan sebelumnya. Penggalan kode proses inisialisasi *claim value* dapat dilihat pada tabel 4.14.

Tabel 4. 14 Penggalan Kode Proses Inisialisasi *Claim Value*

No	Penggalan Kode
1	<code>def inisialisasi_claim_value(self, factor, scores):</code>
2	<code> new_scores = scores</code>
3	<code> for i in range(len(new_scores)):</code>
4	<code> for j in range(len(new_scores[0])):</code>
5	<code> new_scores[i][j] *= factor</code>
6	<code> return new_scores</code>

Langkah berikutnya yaitu perhitungan *trustworthiness score* yang penggalan kodenya dapat dilihat pada Tabel 4.15. Proses menghitung *trustworthiness score* setiap sumber s dilakukan dengan menggunakan persamaan (2.17) yang ditunjukkan pada baris ke-8 dan 9, kemudian dilanjutkan dengan menggunakan persamaan (2.19) yang ditunjukkan pada baris ke-10 dan 11.

Tabel 4. 15 Penggalan Kode Menghitung *Trustworthiness Score*

No	Penggalan Kode
1	<code>def calculate_trustworthiness_score(self, Cs_count_list, new_scores,</code>
2	<code>isAvgLog):</code>
3	<code> trustworthiness_score_list = []</code>
4	<code> for i in range(len(new_scores)):</code>
5	<code> trustworthiness_score = 0</code>
6	<code> for j in range(len(new_scores[i])):</code>
7	<code> trustworthiness_score=trustworthiness_score + new_scores[i][j]</code>
8	<code> if(isAvgLog):</code>
9	<code> trustworthiness_score = (math.log(Cs_count_list[i]))*(trustwor</code>
10	<code>thiness_score/Cs_count_list[i])</code>
11	<code> trustworthiness_score_list.append(trustworthiness_score)</code>
12	<code> for j in range(len(new_scores[i])):</code>
13	<code> if new_scores[i][j] > 0:</code>
14	<code> new_scores[i][j] = trustworthiness_score</code>
15	<code> return trustworthiness_score_list, new_scores</code>
16	
17	
18	
19	

Langkah selanjutnya yaitu perhitungan *belief score* klaim c dengan menggunakan persamaan (2.18). Penggalan kode menghitung *belief score* dapat dilihat pada Tabel 4.16.

Tabel 4. 16 Penggalan Kode Menghitung *Belief Score*

No	Penggalan Kode
1	<code>def calculate_belief_score(self, new_scores):</code>
2	<code> belief_score_list = []</code>
3	<code> for i in range(len(new_scores[0])):</code>
4	<code> belief_score = 0</code>
5	<code> for j in range(len(new_scores)):</code>
6	<code> belief_score = belief_score + new_scores[j][i]</code>
7	
8	<code> belief_score_list.append(belief_score)</code>
9	
10	<code> for j in range(len(new_scores)):</code>
11	<code> if new_scores[j][i] > 0:</code>
12	<code> new_scores[j][i] = belief_score</code>
13	
14	<code> return belief_score_list, new_scores</code>

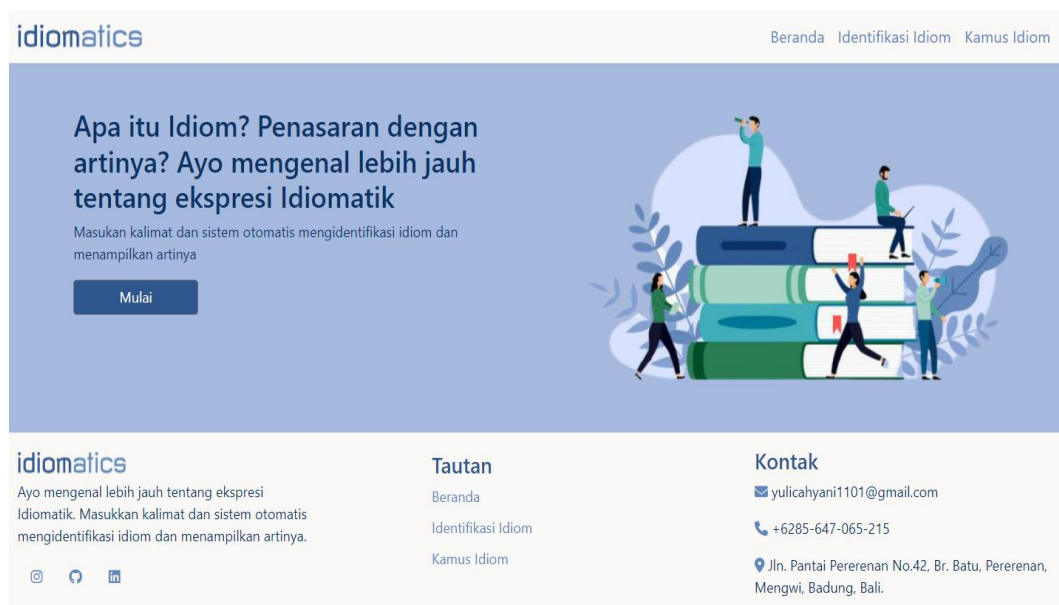
Langkah selanjutnya yaitu proses validasi frasa hasil identifikasi. Penggalan kode proses validasi frasa idiom dapat dilihat pada Tabel 4.17. Proses validasi diawali dengan menghitung nilai parameter *threshold* dengan menggunakan persamaan (3.1) yang ditunjukkan pada baris ke-9. Kemudian *belief score* dari frasa-frasa hasil identifikasi akan dibandingkan dengan parameter *threshold* dimana suatu frasa dianggap benar atau valid sebagai frasa idiom jika mempunyai *belief score* di atas nilai parameter *threshold* yang ditunjukkan pada baris ke-11 sampai baris ke-17. Hasil validasi akan bernilai 1 jika frasa dianggap valid sebagai idiom dan bernilai 0 jika frasad dianggap sebagai bukan idiom.

Tabel 4. 17 Penggalan Kode Proses Validasi Frasa Idiom

No	Penggalan Kode
1	<code>def _predict(self, x):</code>
2	<code> w = x.split(' ')</code>
3	<code> r = w[0] + '(.*)' + w[1]</code>
4	<code> result=self.dataframe[self.dataframe['idiom_claims'].str.contains(</code>
5	<code> r, na=False)]</code>
6	<code> score = result['belief_score'].max()</code>
7	<code> label = 0</code>
8	
9	<code> batas = np.percentile(self.dataframe['belief_score'], q=15)</code>
10	
11	<code> if math.isnan(score) == False:</code>
12	<code> if score > batas:</code>
13	<code> label = 1</code>
14	<code> else:</code>
15	<code> label = 0</code>
16	<code> else:</code>
17	<code> label = 0</code>
18	
19	<code> return label</code>

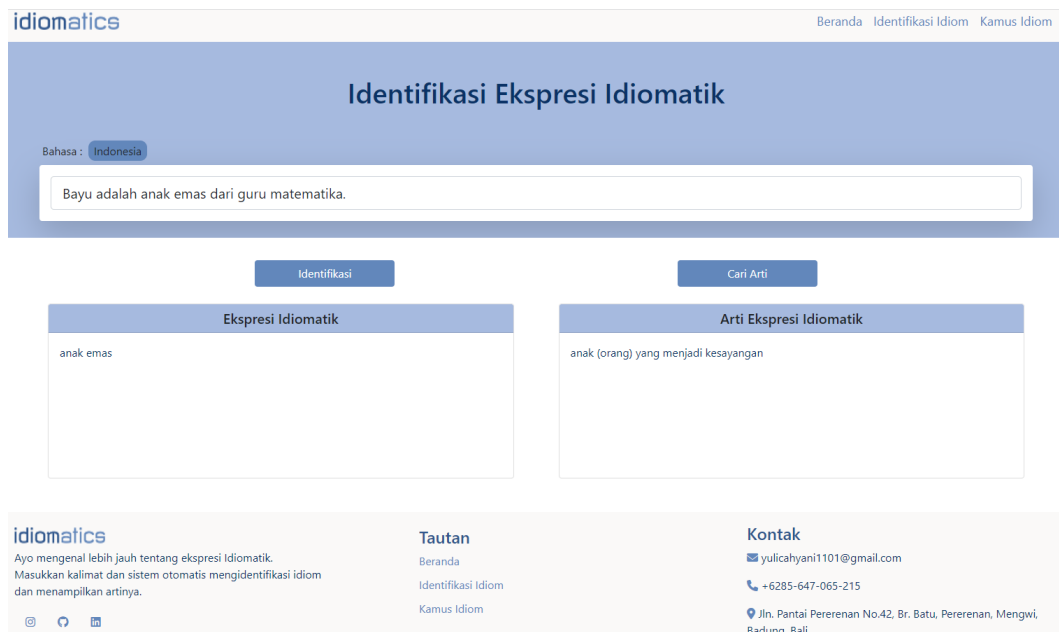
4.1.7 Implementasi Tampilan Antarmuka

Tampilan antarmuka sistem terdiri dari tiga halaman, yaitu halaman beranda, halaman identifikasi idiom, dan halaman kamus idiom. Pada saat pertama kali membuka sistem, akan ditampilkan halaman beranda yang dapat dilihat pada Gambar 4.1. Pada halaman beranda menampilkan nama sistem serta deskripsi singkat dari sistem.



Gambar 4. 1 Halaman Beranda

Halaman identifikasi yang dapat dilihat pada Gambar 4.2. merupakan halaman untuk melakukan identifikasi ekspresi idiomatik. Pada halaman ini terdapat *text box* untuk menerima inputan dari user yang berupa suatu kalimat berpola dasar berbahasa Indonesia. Kemudian terdapat tombol identifikasi untuk melakukan proses identifikasi ekspresi idiomatik pada kalimat inputan dan juga terdapat tombol cari arti untuk menampilkan arti dari ekspresi idiomatik yang telah diidentifikasi sebelumnya.



Gambar 4. 2 Halaman Identifikasi Idiom

Selanjutnya halaman kamus idiom dapat dilihat pada Gambar 4.3. Pada halaman kamus idiom terdapat *text box* untuk menerima inputan dari user yang berupa huruf, kata, atau frasa berbahasa Indonesia. Kemudian terdapat tombol cari untuk melakukan proses pencarian berdasarkan inputan dan menampilkan hasil berupa ekspresi idiomatik beserta arti dan contoh penggunaannya dalam kalimat.



Gambar 4. 3 Halaman Kamus Idiom

4.2 Hasil Penelitian

4.2.1 Validasi Model BERT untuk Klasifikasi Kalimat yang Mengandung Ekspresi Idiomatik

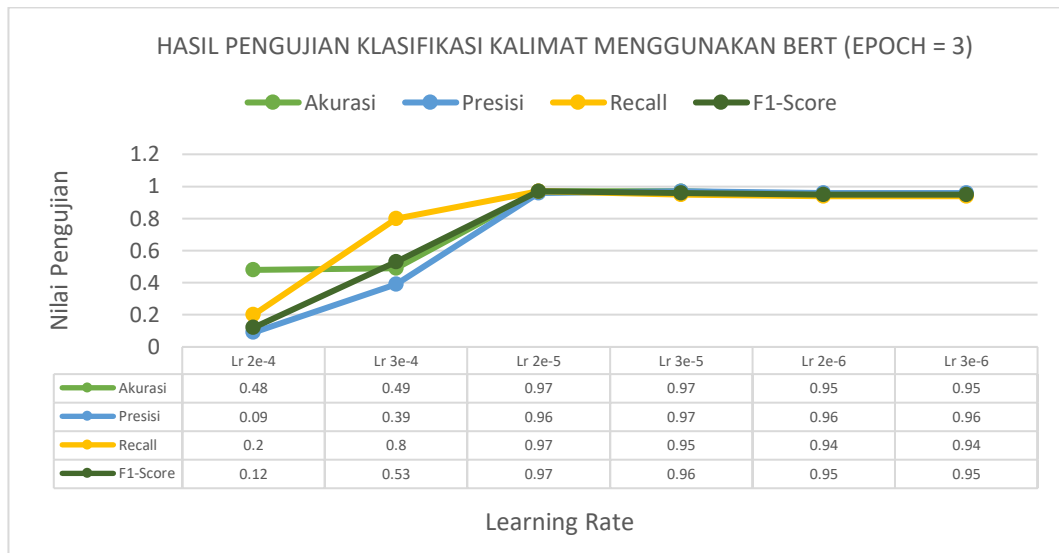
Pada penelitian ini, untuk menentukan model BERT terbaik untuk klasifikasi kalimat berbahasa Indonesia yang mengandung ekspresi idiomatik (kalimat_idiom) atau tidak mengandung ekspresi idiomatik (kalimat_biasa), maka dilakukan *tuning hyperparameter* dengan *K-Fold Cross-Validation*. Metrik ukuran evaluasi yang digunakan adalah akurasi, presisi, *recall*, dan *f1-score*. Pada penelitian ini, menggunakan data kalimat berpola dasar berbahasa Indonesia yang mengandung dan tidak mengandung ekspresi idiomatik. Data kalimat tersebut akan dibagi dengan perbandingan 80% untuk data *training* dan validasi kemudian 20% untuk data *testing*. Validasi model dilakukan dengan menggunakan metode *5-Fold Cross-Validation* dan berdasarkan konfigurasi *hyperparameter* yang dapat dilihat pada Tabel 3.12. Hasil validasi model BERT dapat dilihat pada Tabel 4.18. Pada Tabel 4.18, dapat kita lihat nilai akurasi, presisi, *recall* dan *f1-score* yang didapatkan melalui pengujian dengan *epoch* yaitu 3, 4, 5, 6, dan 7 serta *learning rate* yang digunakan yaitu 2e-4, 3e-4, 2e-5, 3e-5, 2e-6 dan 3e-6. Selanjutnya dari hasil pengujian tersebut akan dianalisis untuk menemukan model BERT untuk klasifikasi kalimat dengan performa terbaik yang selanjutnya akan diuji kembali menggunakan data *testing*.

Pada Tabel 4.18 tersebut dapat kita bandingkan hasil pengujian yang diperoleh dengan menggunakan *epoch* 3, 4, 5, 6, 7. Berdasarkan hasil pengujian dapat kita lihat bahwa pada *learning rate* 2e-4 dan 3e-4 nilai akurasi, presisi, *recall* dan *f1-score* yang diperoleh pada setiap *epoch* cenderung sangat rendah. Sedangkan nilai akurasi, presisi, *recall* dan *f1-score* yang diperoleh pada *learning rate* 2e-5, 3e-5, 2e-6 dan 3e-6 cenderung tinggi dan tidak jauh berbeda satu sama lain. Hal ini terjadi karena pada *learning rate* 2e-4 dan 3e-4, nilai *False Negative* atau *False Positive* yang diperoleh pada tabel *confusion matrix* setiap *fold* yang dapat dilihat pada Lampiran 1, 2, 3, 4 dan 5 cenderung bernilai tinggi, yang berarti banyak kalimat yang memiliki kategori 'kalimat_idiom' diprediksi sebagai 'kalimat_biasa' dan sebaliknya.

Tabel 4. 18 Hasil Validasi Model BERT untuk Klasifikasi Kalimat yang Mengandung Ekspresi Idiomatik

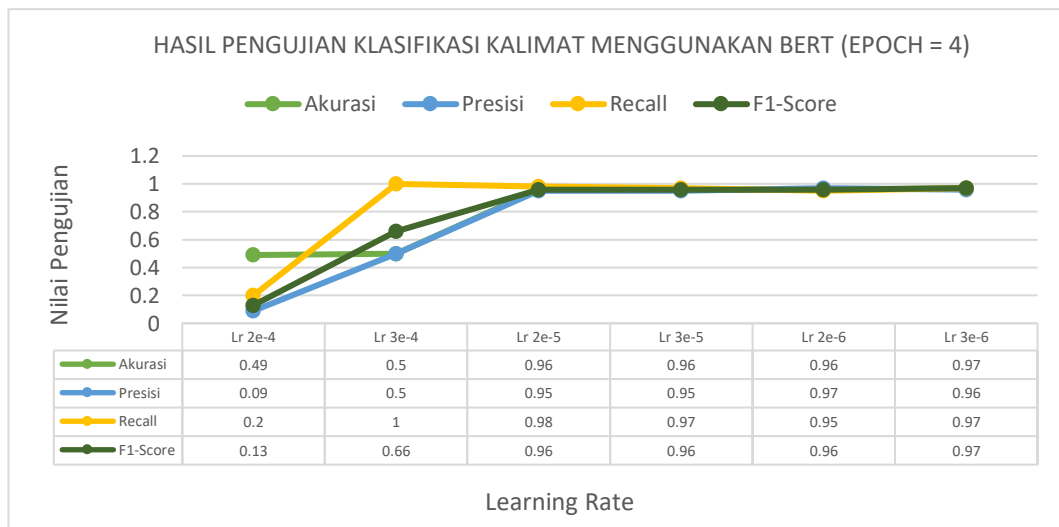
<i>Epoch</i>	<i>Learning Rate</i>	Rata-Rata Hasil Pengujian			
		Akurasi	Presisi	Recall	<i>F1-Score</i>
3	2e-4	0.48	0.09	0.20	0.12
	3e-4	0.49	0.39	0.80	0.53
	2e-5	0.97	0.96	0.97	0.97
	3e-5	0.97	0.97	0.95	0.96
	2e-6	0.95	0.96	0.94	0.95
	3e-6	0.95	0.96	0.94	0.95
4	2e-4	0.49	0.09	0.20	0.13
	3e-4	0.50	0.50	1.0	0.66
	2e-5	0.96	0.95	0.98	0.96
	3e-5	0.96	0.95	0.97	0.96
	2e-6	0.96	0.97	0.95	0.96
	3e-6	0.97	0.96	0.97	0.97
5	2e-4	0.49	0.19	0.4	0.26
	3e-4	0.49	0.49	1.0	0.66
	2e-5	0.98	0.97	0.97	0.97
	3e-5	0.96	0.95	0.96	0.96
	2e-6	0.95	0.95	0.94	0.94
	3e-6	0.97	0.97	0.96	0.97
6	2e-4	0.49	0.09	0.20	0.13
	3e-4	0.50	0.50	1.0	0.66
	2e-5	0.98	0.98	0.99	0.98
	3e-5	0.95	0.96	0.94	0.95
	2e-6	0.95	0.95	0.94	0.95
	3e-6	0.97	0.97	0.97	0.97
7	2e-4	0.49	0.09	0.20	0.13
	3e-4	0.50	0.50	1.0	0.66
	2e-5	0.98	0.98	0.99	0.98
	3e-5	0.96	0.96	0.96	0.96
	2e-6	0.95	0.95	0.95	0.95
	3e-6	0.97	0.97	0.98	0.97

Selanjutnya untuk melihat lebih jelas perbandingan hasil pengujian dari setiap epoch dan learning rate akan dituangkan ke dalam grafik berikut.



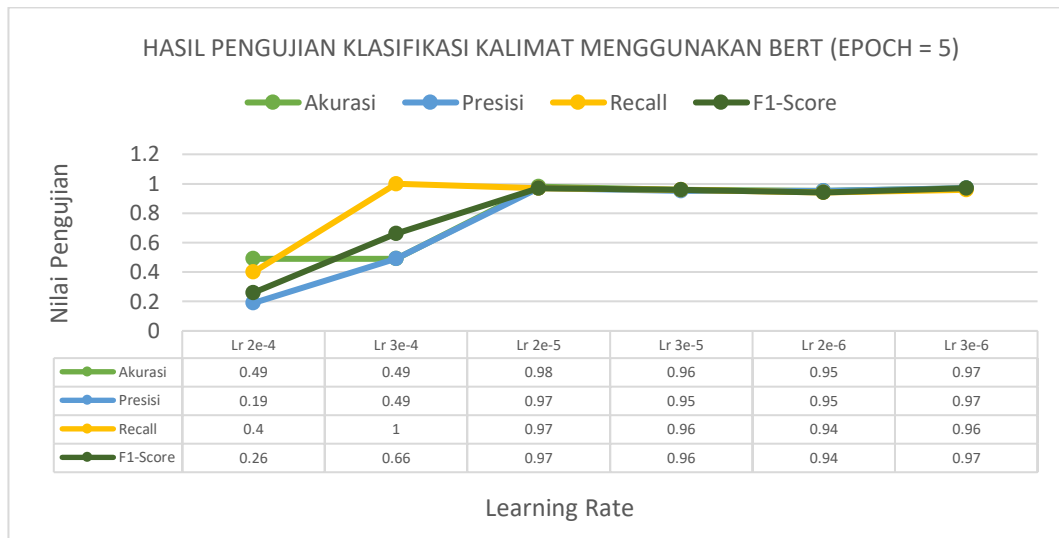
Gambar 4. 4 Grafik Hasil Pengujian Klasifikasi Kalimat *Epoch 3*

Gambar 4.4 menunjukkan perbandingan hasil pengujian pada setiap *learning rate* dengan jumlah *epoch* yaitu 3, dimana pengujian dengan *learning rate* 2e-5 memperoleh hasil tertinggi dengan nilai akurasi, presisi, *recall* dan *f1-score* yaitu 0,97; 0,96; 0,97 dan 0,97.



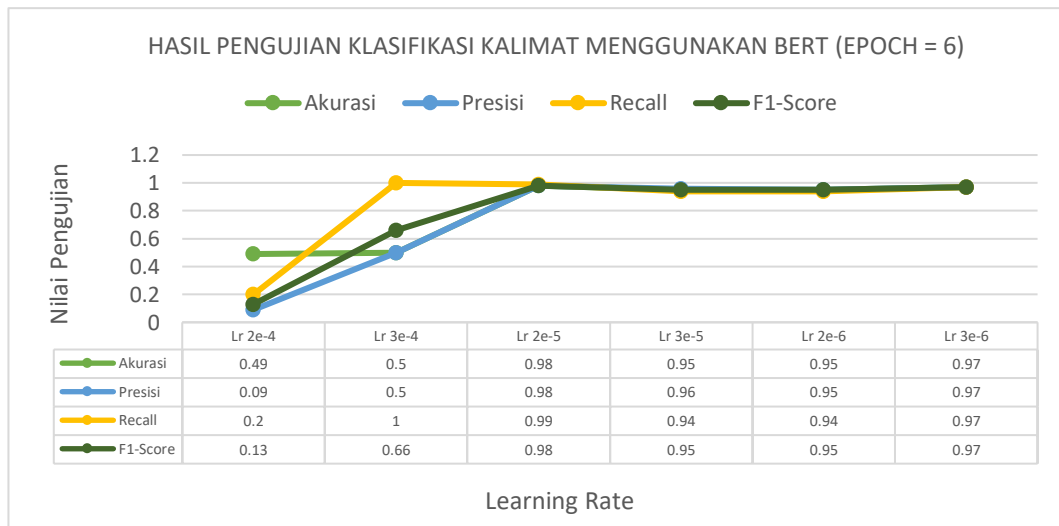
Gambar 4. 5 Grafik Hasil Pengujian Klasifikasi Kalimat *Epoch 4*

Gambar 4.5 menunjukkan perbandingan hasil pengujian pada setiap *learning rate* dengan jumlah *epoch* yaitu 4, dimana pengujian dengan *learning rate* 3e-6 memperoleh hasil tertinggi dengan nilai akurasi, presisi, *recall* dan *f1-score* yaitu 0,97; 0,96; 0,97 dan 0,97.



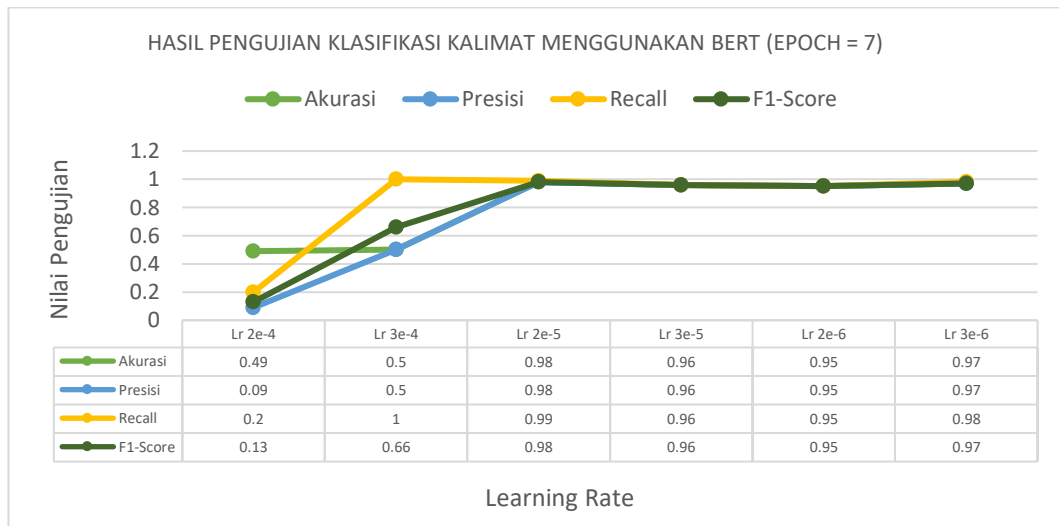
Gambar 4. 6 Grafik Hasil Pengujian Klasifikasi Kalimat *Epoch 5*

Gambar 4.6 menunjukkan perbandingan hasil pengujian pada setiap *learning rate* dengan jumlah *epoch* yaitu 5, dimana pengujian dengan *learning rate* 2e-5 memperoleh hasil tertinggi dengan nilai akurasi, presisi, *recall* dan *f1-score* yaitu 0,98; 0,97; 0,97 dan 0,97.



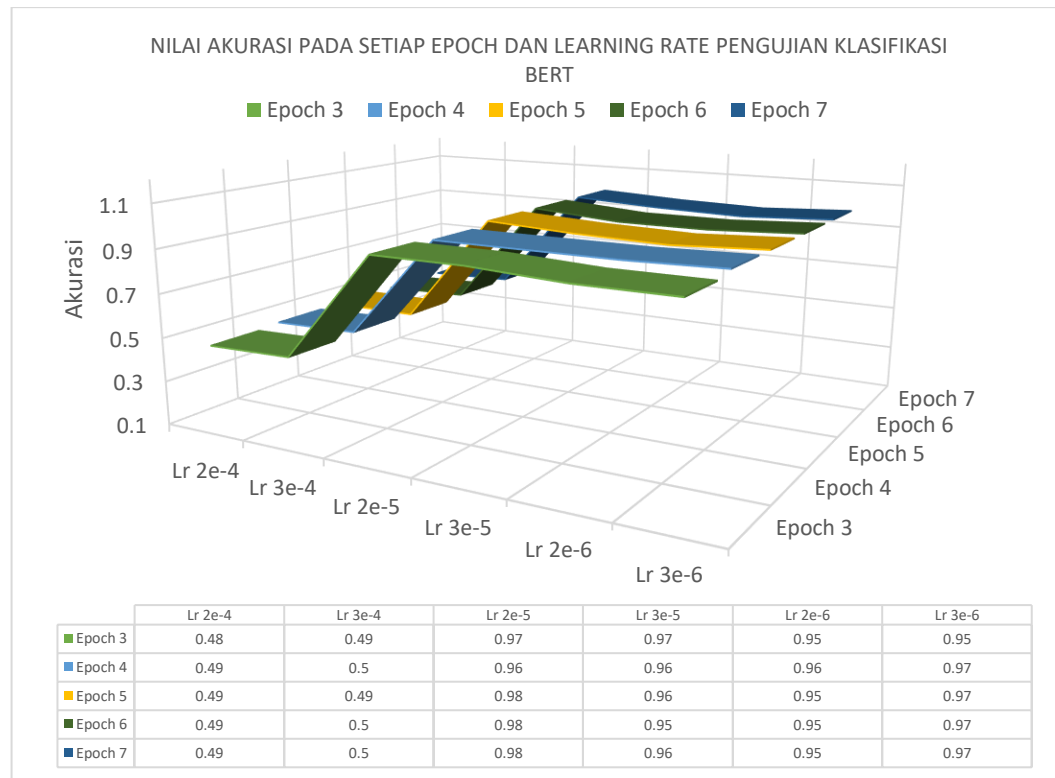
Gambar 4. 7 Grafik Hasil Pengujian Klasifikasi Kalimat *Epoch 6*

Gambar 4.7 menunjukkan perbandingan hasil pengujian pada setiap *learning rate* dengan jumlah *epoch* yaitu 6, dimana pengujian dengan *learning rate* 2e-5 memperoleh hasil tertinggi dengan nilai akurasi, presisi, *recall* dan *f1-score* yaitu 0,98; 0,98; 0,99 dan 0,98.



Gambar 4. 8 Grafik Hasil Pengujian Klasifikasi Kalimat *Epoch 7*

Gambar 4.8 menunjukkan perbandingan hasil pengujian pada setiap *learning rate* dengan jumlah *epoch* yaitu 7, dimana pengujian dengan *learning rate* 2e-5 memperoleh hasil tertinggi dengan nilai akurasi, presisi, *recall* dan *f1-score* yaitu 0,98; 0,98; 0,99 dan 0,98.



Gambar 4. 9 Grafik Nilai Akurasi Pada Setiap *Learning Rate* dan *Epoch*

Gambar 4.9 menunjukkan perbandingan nilai akurasi pada setiap *learning rate* dan *epoch*, dimana pada grafik tersebut dapat dilihat bahwa *learning rate* 2e-5 memperoleh nilai akurasi yang paling optimal.

Berdasarkan analisis yang telah dilakukan pada hasil validasi model dengan *K-Fold Cross-Validation*, dapat dilihat bahwa hasil pengujian pada *learning rate* 2e-5 dengan *epoch* 6 dan 7 memperoleh nilai akurasi, presisi, *recall* dan *f1-score* yang sama dan paling optimal, hal ini menunjukkan bahwa model klasifikasi kalimat menggunakan BERT dengan performa terbaik didapatkan pada *learning rate* 2e-5 dengan minimum *epoch* 6 dengan nilai akurasi, presisi, *recall* dan *f1-score* yaitu 0,98; 0,98; 0,99 dan 0,98.

4.2.2 Evaluasi Model BERT untuk Klasifikasi Kalimat yang Mengandung Ekspresi Idiomatik Menggunakan Data Baru

Pada tahap validasi model dengan *K-Fold Cross-Validation* telah didapatkan model BERT terbaik untuk klasifikasi kalimat yang mengandung ekspresi idiomatik adalah model BERT dengan *hyperparameter* yaitu *learning rate* 2e-5 dan minimum *epoch* 6. Selanjutnya model terbaik ini diuji kembali menggunakan data baru yang belum pernah melewati tahap pelatihan dan validasi sebelumnya. Hasil pengujian menggunakan data baru dapat dilihat pada Tabel 4.19 berikut.

Tabel 4. 19 Hasil Evaluasi dengan Data Baru Model Klasifikasi BERT Terbaik

Pengujian	Akurasi	Presisi	<i>Recall</i>	<i>F1-Score</i>
Training Validasi	0.98	0.98	0.99	0.98
Testing Data Baru	0.98	0.99	0.97	0.98

Dari Tabel 4.19 di atas dapat kita dibandingkan performa model pada saat pelatihan dan validasi dengan performa model saat diuji dengan data baru untuk melihat adanya *overfitting* atau tidak. *Overfitting* adalah suatu keadaan jika model menghasilkan performa yang tinggi pada saat pelatihan dan validasi namun performanya menurun secara signifikan ketika diuji dengan data baru pada proses *testing*. Berdasarkan pada Tabel 4.19, dapat dilihat bahwa model tidak mengalami

overfitting dilihat dari nilai akurasi, presisi, *recall* dan *f1-score* yang dihasilkan saat *testing* data baru yaitu 0,98; 0,99; 0,97; 0,98 tidak jauh berbeda dengan nilai akurasi, presisi, *recall* dan *f1-score* yang dihasilkan pada *training* dan validasi.

4.2.3 Evaluasi Identifikasi Ekspresi Idiomatik Menggunakan *Truth*

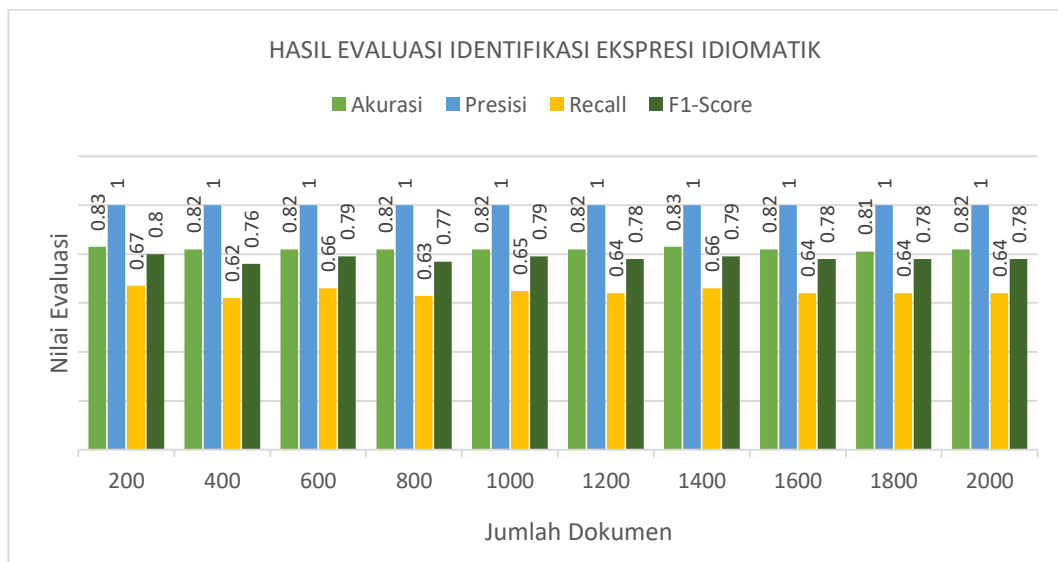
Discovery

Pada penelitian ini, untuk mengetahui performa identifikasi ekspresi idiomatik menggunakan *Truth Discovery* dilakukan pengujian atau evaluasi untuk mendapatkan nilai dari akurasi, presisi, *recall*, dan *f1-score*. Pada pengujian ini, seluruh data kalimat Bahasa Indonesia digunakan sebagai data uji, hal ini terjadi karena dalam identifikasi yang tidak memerlukan data latih. Pengujian akan dilakukan sebanyak 10 kali dengan jumlah data uji yang berbeda-beda dalam setiap pengujian. Selanjutnya hasil evaluasi didapatkan melalui rata-rata dari 10 kali pengujian tersebut. Hasil evaluasi identifikasi ekspresi idiomatik menggunakan *Truth Discovery* dapat dilihat pada Tabel 4.20.

Tabel 4. 20 Hasil Evaluasi Identifikasi Ekspresi Idiomatik

Pengujian	Jumlah Data	Akurasi	Presisi	<i>Recall</i>	<i>F1-Score</i>
1	200	0.83	1.0	0.67	0.80
2	400	0.82	1.0	0.62	0.76
3	600	0.82	1.0	0.66	0.79
4	800	0.82	1.0	0.63	0.77
5	1000	0.82	1.0	0.65	0.79
6	1200	0.82	1.0	0.64	0.78
7	1400	0.83	1.0	0.66	0.79
8	1600	0.82	1.0	0.64	0.78
9	1800	0.81	1.0	0.64	0.78
10	2000	0.82	1.0	0.64	0.78
Rata-rata		0.82	1.0	0.64	0.78

Pada Tabel 4.20 di atas dapat kita lihat nilai akurasi, presisi, *recall* dan *f1-score* yang diperoleh dari pengujian dengan jumlah data 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800 dan 2000. Perbandingan hasil evaluasi setiap pengujian dapat dilihat pada Gambar 4.6.



Gambar 4. 10 Grafik Hasil Evaluasi Identifikasi Ekspresi Idiomatik

Gambar 4.9 menunjukkan perbandingan hasil evaluasi pada setiap jumlah data. Berdasarkan Gambar 4.9 tersebut, nilai akurasi, presisi, *recall* dan *f1-score* yang diperoleh pada setiap jumlah data tidak jauh berbeda dan tidak terdapat perubahan yang besar. Nilai akurasi yang dihasilkan pada setiap jumlah data berkisaran pada angka 0,81 sampai 0,83. Kemudian untuk nilai presisi pada setiap jumlah data memiliki nilai yang sama yaitu 1,0. Hal ini terjadi karena pada *confusion matrix* yang dapat dilihat pada Lampiran 6 nilai *False Positive* yang diperoleh pada setiap pengujian bernilai 0, yang berarti tidak ada hasil identifikasi yang memiliki kategori ‘bukan_idiom’ diprediksi sebagai ‘idiom’. Selanjutnya untuk nilai *recall* pada setiap jumlah data cukup rendah berkisaran pada angka 0,62 sampai 0,66. Hal ini terjadi karena pada *confusion matrix* nilai *False Negative* yang diperoleh pada setiap pengujian cukup tinggi, yang berarti jumlah hasil identifikasi yang memiliki kategori ‘idiom’ diprediksi sebagai ‘bukan_idiom’ cukup banyak. Kemudian nilai *f1-score* yang dihasilkan pada setiap jumlah data berkisaran pada angka 0,76 sampai 0,80.

Dari pengujian yang telah dilakukan, dapat dikatakan model identifikasi ekspresi idiomatik menggunakan *Truth Discovery* memiliki performa cukup baik dengan rata-rata nilai akurasi, presisi, *recall* dan *f1-score* yang diperoleh yaitu 0,82; 1,0; 0,64 dan 0,78. Rata-rata akurasi, presisi dan *f1-score* memiliki nilai yang cukup

baik sedangkan nilai *recall* yang dihasilkan cukup rendah, yang berarti jumlah ekspresi idiomatik yang tidak berhasil diidentifikasi cukup banyak. Hal tersebut dapat disebabkan oleh beberapa faktor sebagai berikut:

- a. Pada tahap klasifikasi, BERT tidak berhasil mengklasifikasikan kalimat ke dalam kategori *kalimat_idiom*.
- b. Prediksi kelas kata yang dihasilkan pada tahap *Part-of-Speech Tagging* kurang optimal sehingga menyebabkan proses *Chunking* juga memberikan hasil yang kurang optimal dalam menemukan frasa kandidat yang memiliki konstruksi sintaksis pembentuk ekspresi idiomatik.
- c. Keterbatasan data sumber web, dimana klaim berupa ekspresi idiomatik yang disediakan oleh sumber web cenderung sedikit sehingga pada proses identifikasi terdapat beberapa frasa yang sebenarnya merupakan idiom tapi tidak dinyatakan sebagai idiom karena tidak ada klaim dari sumber web atas frasa tersebut.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pada penelitian yang telah dilakukan serta hasil yang diperoleh, dapat ditarik kesimpulan sebagai berikut:

1. Hasil validasi model menggunakan *5-Fold Cross Validation* menunjukkan bahwa model *Bidirectional Encoder Representations from Transformers* (BERT) terbaik untuk klasifikasi kalimat berbahasa Indonesia yang mengandung ekspresi idiomatik adalah model BERT dengan *hyperparameter* yaitu *learning rate* $2e-5$ dan minimum *epoch* 6. Hasil evaluasi dengan data baru menunjukkan performa yang baik dengan nilai akurasi sebesar 0,98; presisi sebesar 0,99; *recall* sebesar 0,97 dan *f1-score* sebesar 0,98.
2. Hasil 10 kali pengujian dengan jumlah data yang berbeda menunjukkan bahwa identifikasi ekspresi idiomatik menggunakan *Truth Discovery* memiliki performa yang cukup baik dalam mengidentifikasi ekspresi idiomatik pada kalimat berbahasa Indonesia dengan rata-rata akurasi sebesar 0,82; presisi sebesar 1,0; *recall* sebesar 0,64 dan *f1-score* sebesar 0,78.

5.2 Saran

Dari penelitian yang telah dilakukan serta hasil yang diperoleh, saran-saran yang dapat disampaikan untuk dapat dipertimbangkan dalam pengembangan dari penelitian selanjutnya adalah sebagai berikut:

1. Algoritma HMM untuk *Part-of-Speech Tagging* dapat diganti dengan algoritma lainnya seperti *Brill Tagger* yang menggunakan aturan leksikal dan kontekstual berdasarkan *Transformation Based Learning* untuk mendapatkan hasil prediksi kelas kata yang lebih optimal.

2. Memperbanyak data sumber web yang digunakan dalam membangun model *Truth Discovery* sehingga dalam identifikasi ekspresi idiomatik memperoleh akurasi, presisi, *recall* dan *f1-score* yang lebih baik.

DAFTAR PUSTAKA

- Anggito, A., & Setiawan, J. (2018.) *Metodelogi Penelitian Kualitatif*. CV Jejak: Jawa Barat.
- Arman, A. A., Putra, N., A. B., Purwarianti, A., Kuspriyanto. (2013). Syntactic Phrase Chunking for Indonesian Language. *Procedia Technology*, 11, 635-640.
- Astuti, Purwani I. (2013). Analisis Terjemahan Idiom dalam Buku the Secret Karya Rhonda Byrne. *Jurnal Widyatama*, 22(1), 23–37
- Barrera, A., Verma, R., & Vincent, R. (2011). SemQuest: University of Houston's Semantics-based Question Answering System. *Nist.Gov, November 2011*.
- Berti-Equille, L., and Borge-Holthoefer, J. (2015). Veracity of data: From truth discovery computation algorithms to models of misinformation dynamics. *Synthesis Lectures on Data Management*, 7(3), 1-155.
- Chaer, Abdul. (1997). *Kamus Ungkapan Bahasa Indonesia*. Jakarta: Rineka Cipta.
- Chaer, Abdul. (2013). *Pengantar Semantik Bahasa Indonesia*. Jakarta: Rineka Cipta.
- Dinakaramani, A., Rashel, F., Luthfi, A., & Manurung, R. (2014). Designing an Indonesian Part of Speech Tagset and Manually Tagged Indonesian Corpus. *Proceedings of the International Conference on Asian Language Processing 2014, IALP 2014*, 66-69.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies- Proceedings of the Conference*, 1(Mlm), 4171–4186.
- Fang, Xiu S., Sheng, Quan Z., Wang, X., Zhang, Wei E., Ngu, Anne H.H. (2017). From Appearance to Essence: Comparing Truth Discovery Methods without Using Ground Truth. *In Proceedings of ACM Conference*, Washington, DC, USA, July 2017 (Conference'17), 11.
- Hurford, J.M and Heasley, B. (2007). *Semantic: A Course book*. Cambridge: Cambridge University.
- Jurafsky, D., and Martin, James H. (2017). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Third Edition*.
- Kachuee, S., & Sharifkhani, M. (2022). *TiltedBERT: Resource Adjustable Version of BERT*. <http://arxiv.org/abs/2201.03327>
- Khak, M. A. (2011). Idiom Dalam Bahasa Indonesia: Struktur Dan Maknai. *Widyaparwa*, 39(2), 141–154.
- Kula, S., Kozik, R., & Choraś, M. (2021). Implementation of the BERT-derived architectures to tackle disinformation challenges. *Neural Computing and Applications*. <https://doi.org/10.1007/s00521-021-06276-0>
- Li, Y., Gao, J., Meng, C., Li, Q., & Su, L. (2016). A Survey on Truth Discovery. *ACM Sigkdd Explorations Newsletter*, 17 (2), 1-16.

- Mao, R., Lin, C., Guerin, F. (2018). Word Embedding and WordNet Based Metaphor Identification and Interpretation. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 1–10.
- Moeliono, Anton M., Lapoliwa, H., Alwi, H., Sasangka, S. S. T. W., Sugiyono. (2017). *Tata Bahasa Baku Bahasa Indonesia Edisi Keempat*. Jakarta: Badan Pengembangan dan Pembinaan Bahasa.
- Munika, M., Shaky, S., & Shrestha, A. (2019). *Fine-grained Sentiment Classification using BERT*. *International Conference on Artificial Intelligence for Transforming Business and Society, AITB 2019*, 1, 1–5.
- Pasternack, J., & Roth, D. (2010). Knowing What to Believe (when you already know something). *Coling 2010 - 23rd International Conference on Computational Linguistics, Proceedings of the Conference*, 2(August), 877–885.
- Pressman, R. S. (2012). *Rekayasa Perangkat Lunak Pendekatan Praktisi Edisi 7*. Yogyakarta: ANDI.
- Ramadhanti, F., Wibisono, Y., Sukanto, Rosa A. (2019). Analisis Morfologi untuk Menangani Out-of-Vocabulary Words pada Part-of-Speech Tagger Bahasa Indonesia Menggunakan Hidden Markov Model. *Jurnal Linguistik Komputasional*, 2(1), Maret 2019.
- Ravichandiran, S. (2021). *Getting started with Google BERT: Build and train state-of-the-art natural language processing models using BERT*. Birmingham: Packt Publishing.
- Rothman, D. (2021). *Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP & Python, PyTorch, TensorFlow, BERT, RoBERTa & more*. Birmingham: Packt Publishing.
- Sanjaya, N. A., Abdessalem, T., Ba, M. L., & Bressan, S. (2018). Harnessing Truth Discovery Algorithms on The Topic Labelling Problem. *In Proceedings of the 20th International Conference on Information Integration and Web-based Applications & Services*, 8-14.
- Sarkar, Dipanjam. (2019). *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing, Second Edition*. Bangalore, Karnataka, India: Appress Media.
- Shutova, E., Kiela, D., Maillard, J. (2016). Black Holes and White Rabbits: Metaphor Identification with Visual Features. *Proceedings of NAACL-HLT 2016*, 160–170.
- Srinivasa-Desikan, B. (2018). *Natural Language Processing and Computational Linguistics*. Birmingham: Packt Publishing Ltd.
- Steven Bird, Ewan Klein, Edward Loper. (2009). *Natural Language Processing with Python*. United States of America: O'Reilly Media.
- Susanto, R., & Andriana, A. D. (2016). Perbandingan Model Waterfall dan Prototyping untuk Pengembangan Sistem Informasi. *Majalah Ilmiah UNIKOM*, 14(1), 41–46. <https://doi.org/10.34010/miu.v14i1.174>
- Syahroni, Abd. W., and Harsono. (2019). Aplikasi Penentuan Kategori dan Fungsi Sintaksis Kalimat Bahasa Indonesia. *Jurnal Nasional Informatika dan Teknologi Jaringan*, 4(1).

- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 2017-Decem (Nips)*, 5999–6009.
- Verma, R., and Vuppuluri, V. (2015). A New Approach for Idiom Identification Using Meanings and the Web*. *Proceedings of Recent Advances in Natural Language Processing*, 681–687.
- Virdaus, Varia V. (2020). Ekspresi Idiomatik dalam Lirik “Nine Track Mind” Charlie Puth Album 2016. *Media of Teaching Oriented and Children*, 4(1).
- Widjono. (2007). *Bahasa Indonesia*. Jakarta: PT Grasindo.
- Zayed, O., McCrae, John P., Buitelaar, P. (2018). Phrase-Level Metaphor Identification using Distributed Representations of Word Meaning. *Proceedings of the Workshop on Figurative Language Processing*, 81–90.
- Zhao, S., Pascual, D., Brunner, G., & Wattenhofer, R. (2021). Of Non-Linearity and Commutativity in BERT. *Proceedings of the International Joint Conference on Neural Networks, 2021-July*.

LAMPIRAN

Lampiran 1. *Confusion Matrix* untuk Pengujian Klasifikasi BERT *Epoch 3*

<i>Epoch</i>	<i>Learning Rate</i>	<i>Fold</i>	<i>Confusion Matrix</i>			
			TN	FP	FN	TP
3	2e-4	1	0	175	0	145
		2	161	0	159	0
		3	158	0	162	0
		4	152	0	168	0
		5	155	0	165	0
	3e-4	1	0	175	0	145
		2	0	161	0	159
		3	158	0	162	0
		4	0	152	0	158
		5	0	155	0	165
	2e-5	1	156	19	10	135
		2	156	5	5	154
		3	155	3	1	161
		4	152	0	3	165
		5	155	0	0	165
	3e-5	1	169	6	19	126
		2	157	4	7	152
		3	155	3	0	152
		4	151	1	2	166
		5	154	1	3	162
	2e-6	1	156	19	19	126
		2	158	3	18	141
		3	154	4	0	162
		4	152	0	2	166
		5	154	1	1	164
	3e-6	1	151	24	18	127
		2	157	4	16	143
		3	156	2	1	161
		4	152	0	2	166
		5	155	0	2	163

Lampiran 2. *Confusion Matrix* untuk Pengujian Klasifikasi BERT *Epoch 4*

<i>Epoch</i>	<i>Learning Rate</i>	<i>Fold</i>	<i>Confusion Matrix</i>			
			TN	FP	FN	TP
4	2e-4	1	0	161	0	159
		2	162	0	158	0
		3	160	0	160	0
		4	154	0	166	0
		5	161	0	159	0
	3e-4	1	0	161	0	159
		2	0	162	0	158
		3	0	160	0	160
		4	0	158	0	166
		5	0	161	0	159
	2e-5	1	124	37	3	156
		2	155	7	2	156
		3	160	0	4	156
		4	154	0	1	165
		5	161	0	0	159
	3e-5	1	130	31	3	156
		2	158	4	5	153
		3	157	3	7	153
		4	154	0	4	162
		5	161	0	1	158
	2e-6	1	146	15	18	141
		2	157	5	8	150
		3	160	0	6	154
		4	154	0	0	166
		5	161	0	2	157
	3e-6	1	139	22	9	150
		2	156	6	4	154
		3	160	0	2	158
		4	154	0	0	166
		5	161	0	1	158

Lampiran 3. *Confusion Matrix* untuk Pengujian Klasifikasi BERT *Epoch 5*

<i>Epoch</i>	<i>Learning Rate</i>	<i>Fold</i>	<i>Confusion Matrix</i>			
			TN	FP	FN	TP
5	2e-4	1	0	175	0	145
		2	161	0	159	0
		3	158	0	162	0
		4	0	152	0	168
		5	155	0	165	0
	3e-4	1	0	175	0	145
		2	0	161	0	159
		3	0	158	0	162
		4	0	152	0	168
		5	0	155	0	165
	2e-5	1	166	9	5	140
		2	157	4	6	153
		3	156	2	3	159
		4	152	0	2	166
		5	154	1	0	165
	3e-5	1	169	6	13	132
		2	154	7	9	150
		3	141	17	0	162
		4	148	4	2	166
		5	154	1	1	164
	2e-6	1	144	31	15	130
		2	157	4	16	143
		3	155	3	5	157
		4	152	0	1	167
		5	154	1	2	163
	3e-6	1	162	13	17	128
		2	158	3	6	153
		3	158	0	1	161
		4	152	0	1	167
		5	155	0	1	164

Lampiran 4. *Confusion Matrix* untuk Pengujian Klasifikasi BERT *Epoch 6*

<i>Epoch</i>	<i>Learning Rate</i>	<i>Fold</i>	<i>Confusion Matrix</i>			
			TN	FP	FN	TP
6	2e-4	1	0	161	159	0
		2	162	0	158	0
		3	160	0	160	0
		4	154	0	166	0
		5	161	0	159	0
	3e-4	1	0	161	0	159
		2	0	162	0	158
		3	0	160	0	160
		4	0	154	0	166
		5	0	161	0	159
	2e-5	1	150	11	4	155
		2	161	1	1	155
		3	160	0	0	160
		4	154	0	0	166
		5	161	0	1	158
	3e-5	1	140	21	6	153
		2	160	2	19	139
		3	156	4	11	149
		4	151	3	4	162
		5	158	3	0	159
	2e-6	1	133	28	13	146
		2	157	5	19	139
		3	159	1	9	151
		4	154	0	2	164
		5	161	0	1	158
	3e-6	1	142	19	8	151
		2	159	3	9	149
		3	160	0	2	158
		4	153	1	1	165
		5	161	0	1	158

Lampiran 5. *Confusion Matrix* untuk Pengujian Klasifikasi BERT *Epoch 7*

<i>Epoch</i>	<i>Learning Rate</i>	<i>Fold</i>	<i>Confusion Matrix</i>			
			TN	FP	FN	TP
7	2e-4	1	0	161	0	159
		2	162	0	158	0
		3	160	0	160	0
		4	154	0	166	0
		5	161	0	159	0
	3e-4	1	0	161	0	159
		2	0	162	0	158
		3	0	160	0	160
		4	0	154	0	166
		5	0	161	0	166
	2e-5	1	149	12	1	158
		2	159	3	0	158
		3	159	1	1	159
		4	154	0	1	165
		5	161	0	1	158
	3e-5	1	14	21	8	151
		2	157	5	1	157
		3	159	1	9	151
		4	152	2	6	160
		5	161	0	1	158
	2e-6	1	130	31	14	145
		2	157	5	13	145
		3	159	1	9	151
		4	154	0	2	164
		5	161	0	1	158
	3e-6	1	143	18	8	151
		2	157	5	1	157
		3	160	0	2	158
		4	154	0	1	165
		5	161	0	1	158

Lampiran 6. *Confusion Matrix* untuk Pengujian Identifikasi *Truth Discovery*

Pengujian	Jumlah Data	TN	FP	FN	TP
1	200	100	0	33	67
2	400	211	0	71	118
3	600	293	0	104	203
4	800	413	0	141	246
5	1000	480	0	178	342
6	1200	598	0	211	391
7	1400	696	0	238	466
8	1600	799	0	326	586
9	1800	888	0	3226	586
10	2000	1000	0	358	642