SKRIPSI

PERBANDINGAN NORMALISASI KATA *SLANG* BAHASA INDONESIA MENGGUNAKAN MODEL *FASTTEXT* & *WORD2VEC* DENGAN PENDEKATAN *NATURAL LANGUAGE PROCESSING*

OLEH
Rifqah Nur Surayya M.Jen
07352011097

PROGAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS KHAIRUN
TERNATE

2024

LEMBAR PENGESAHAN

PERBANDINGAN NORMALISASI KATA *SLANG* BAHASA INDONESIA MENGGUNAKAN MODEL *FASTTEXT* DAN *WORD2VEC* DENGAN PENDEKATAN *NATURAL LANGUAGE PROCESSING*

Oleh
Rifqah Nur Surayya M. Jen
07352011097

Skripsi ini telah disahkan
Tanggal 31 Mei 2024

Menyetujui
Tim Penguji

Ketua Penguji

Dr. ASSAF ARIEF, S.T., M.Eng.
NIP. 198307102008121001

Pembimbing I

MUHAMMAD FHADLI, S.Kom., M.Sc.
NIP. 19961123202311012

Anggota Penguji

Ir. AMAL KHAIRAN, S.T., M.Eng., IPM.
NIP. 197401112003121003

Pembimbing II

SYARIFUDDIN N. KAPITA, S.Pd., M.Si.
NIP. 199103122024211001

Anggota Penguji

YASIR MUIN, S.T., M.Kom.
NIDN. 9990582796

Mengetahui/Menyetujui

Koordinator Program Studi
Informatika

ROSIHAN, S.T., M.Cs.
NIP. 197607192010121001

Dekan Fakultas Teknik
Universitas Khairun

Ir. ENDAH HARISUN, S.T., M.T., CRP.
NIP. 197511302005011013

LEMBAR PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: Rifqah Nur Surayya M.Jen
NPM	: 07352011097
Fakultas	: Teknik
Jurusan/Program Studi	: Informatika
Judul Skripsi	: Perbandingan Normalisasi Kata *Slang* Bahasa Indonesia Menggunakan Model *FastText* & *Word2Vec* Dengan Pendekatan *Natural Language Processing*

Dengan ini menyatakan bahwa penulisan Skripsi yang telah saya buat ini merupakan hasil karya sendiri dan benar keasliannya. Apabila ternyata di kemudian hari penulisan Skripsi ini merupakan hasil plagiat atau penjiplakan terhadap karya orang lain, maka saya bersedia mempertanggung jawabkan sekaligus bersedia menerima sanksi berdasarkan aturan tata tertib di Universitas Khairun.

Demikian pernyataan ini saya buat dalam keadaan sadar dan tidak dipaksakan.

Rifqah Nur Surayya M.Jen

HALAMAN PERSEMBAHAN

Dengan rasa syukur yang mendalam, dengan telah diselesaikannya Skripsi ini Penulis mempersembahkannya kepada:

1. Kepada kedua orang tua yang telah memberikan segenap cinta dan kasih sayang, serta keluarga besar penulis yang selalu memberikan doa terbaik bagi penulis.
2. Kepada *my best partner* yaitu pemilik NPM "07351611044" yang telah berkontribusi banyak dalam penulisan skripsi ini, meluangkan baik tenaga, waktu, pikiran maupun materi. Telah menjadi rumah, pendamping dalam segala hal yang menemani, mendukung ataupun menghibur dalam kesedihan, mendengar keluh kesah, memberi semangat untuk pantang menyerah. Terima kasih telah menjadi bagian awal dari perjalanan kuliah penulis hingga sekarang.
3. Kepada sahabat penulis sejak SMP, yaitu Suci Wahyuni Ilyas, Asrifany Irawaty, Rahayu Hidayat dan Nurmahanty Admin yang selalu mendengarkan keluh kesah, menjadi tempat cerita dan memberikan dukungan penuh kepada penulis.
4. Kepada sahabat penulis sejak SMA, yaitu Nora, Wiwit, Dika, Didin, Fitri, Fadila, Sabina, Arissa, Zibran, Hamam, Alif, Farhan, Galang, Akmal yang selalu memberikan semangat kepada penulis.
5. Kepada sahabat penulis di bangku perkuliahan yang selalu membersamai dalam empat tahun ini, yaitu Nur Hanifa Lasae, Ismiarsih Suadewista Duwila, Rahmat Kalfi, M. Rezal Karyanto dan Muh. Irfan Mansyur yang selalu siap sedia di saat penulis butuh dan tak pernah henti saling menyemangati.
6. Kepada sahabat GenBI yang telah menemani selama hampir dua tahun, yaitu Nurmala Sari Abd. Rahman, Rohid S. Arahman, Niswah Hilmaniah Husain, Rahmad S. Arahman, Chichi Nurul Maryam, Khalid Bin Wahid Fatcepon, Nurul Anggraini Faizal dan Ryant Wisudahidawan U. Omo yang senantiasa memberikan motivasi untuk menjadi lebih baik.
7. *Last but no least, I wanna thank me, I wanna thank me for believing in me, I wanna thank me for doing all this hard work, I wanna thank me for having no days off, I wanna thank me for never quitting, I wanna thank me for just being me at all times.*

KATA PENGANTAR

Puji syukur saya panjatkan kepada Allah *Subhanahu wata'ala* yang telah melimpahkan rahmat, taufik, hidayah-Nya. Sehingga penulis dapat menyusun skripsi ini dengan judul "Perbandingan Normalisasi Kata *Slang* Bahasa Indonesia menggunakan Model *FastText* & *Word2Vec* dengan Pendekatan *Natural Language Processing*", ini dapat diselesaikan untuk memenuhi salah satu persyaratan penyelesaian pendidikan sarjana Informatika Strata Satu (S1) pada Program Studi Informatika Fakultas Teknik Universitas Khairun.

Untuk menyelesaikan skripsi ini penulis sepenuhnya mendapat dukungan dari banyak pihak, oleh karena itu dengan rendah hati penulis mengucapkan terimakasih kepada:

1. Bapak Dr. Ridha Ajam, M.Hum., selaku Rektor Universitas Khairun Ternate.
2. Bapak Endah Harisun,S.T., M.T., selaku dekan Fakultas Teknik Universitas Khairun.
3. Bapak Rosihan, S.T., M.Cs., selaku Ketua Program Studi Informatika Fakultas Teknik Universitas Khairun.
4. Bapak Muhammad Fhadli, S.Kom., M.Sc., sebagai Pembimbing I yang telah bersedia meluangkan waktunya untuk memberikan arahan serta bimbingannya selama penulis melakukan penelitian dan penyelesaian skripsi ini.
5. Bapak Syarifuddin N. Kapita, S.Pd., M.Si., sebagai Pembimbing II yang telah bersedia meluangkan waktunya untuk memberikan bantuan dan bimbingan serta arahan kepada penulis dalam pengerjaan skripsi ini.
6. Bapak Dr. Assaf Arief, S.T., M.Eng., selaku Penguji I yang telah memberikan masukan, kritikan, dan saran serta dorongan dalam penyelesaian skripsi ini.
7. Bapak Ir. Amal Khairan, S.T., M.Eng., IPM., selaku Penguji II yang telah memberikan masukan, kritikan, dan saran serta dorongan dalam penyelesaian skripsi ini.
8. Bapak Yasir Muin, S.T., M.Kom., selaku Penguji III yang telah memberikan masukan, kritikan, dan saran serta dorongan dalam penyelesaian skripsi ini.
9. Kepada kedua orang tua yang telah memberikan segenap cinta dan kasih sayang, serta keluarga besar penulis yang selalu memberikan doa terbaik bagi penulis.
10. Kepada *my best partner* yaitu pemilik NPM "07351611044" yang telah berkontribusi banyak dalam penulisan skripsi ini, meluangkan baik tenaga, waktu, pikiran maupun materi. Telah menjadi rumah, pendamping dalam segala hal yang menemani,

mendukung ataupun menghibur dalam kesedihan, mendengar keluh kesah, memberi semangat untuk pantang menyerah. Terima kasih telah menjadi bagian awal dari perjalanan kuliah penulis hingga sekarang.

11. Kepada sahabat penulis sejak SMP, yaitu Suci Wahyuni Ilyas, Asrifany Irawaty, Rahayu Hidayat dan Nurmahanty Admin yang selalu mendengarkan keluh kesah, menjadi tempat cerita dan memberikan dukungan penuh kepada penulis.
12. Kepada sahabat penulis sejak SMA, yaitu Nora, Wiwit, Dika, Didin, Fitri, Fadila, Sabina, Arissa, Zibran, Hamam, Alif, Farhan, Galang, Akmal yang selalu memberikan semangat kepada penulis.
13. Kepada sahabat penulis di bangku perkuliahan yang selalu membersamai dalam empat tahun ini, yaitu Nur Hanifa Lasae, Ismiarsih Suadewista Duwila, Rahmat Kalfi, M. Rezal Karyanto dan Muh. Irfan Mansyur yang selalu siap sedia di saat penulis butuh dan tak pernah henti saling menyemangati.
14. Kepada sahabat GenBI yang telah menemani selama hampir dua tahun, yaitu Nurmala Sari Abd. Rahman, Rohid S. Arahman, Niswah Hilmaniah Husain, Rahmad S. Arahman, Chichi Nurul Maryam, Khalid Bin Wahid Fatcepon, Nurul Anggraini Faizal dan Ryant Wisudahidawan U. Omo yang senantiasa memberikan motivasi untuk menjadi lebih baik.

Akhir kata, dengan penuh kesadaran diri dan segala kerendahan hati penulis menyadari hanya Allah yang memiliki segala kesempurnaan, sehingga masih banyak lagi rahasia-Nya yang belum tergali dan penulis ketahui. Semoga Allah *Subhanahu wata'ala* selalu memberikan yang terbaik kepada semua pihak yang selalu membantu penulis. Aamiin.

Ternate, 30 Mei 2024

Penulis

DAFTAR ISI

Halaman

BAB III METODE PENELITIAN

BAB IV HASIL DAN PEMBAHASAN

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

ABSTRAK

PERBANDINGAN NORMALISASI KATA *SLANG* BAHASA INDONESIA MENGGUNAKAN MODEL *FASTTEXT* & *WORD2VEC* DENGAN PENDEKATAN *NATURAL LANGUAGE PROCESSING*

Rifqah Nur Surayya M.Jen[1], Muhammad Fhadli[2] , Syarifuddin N. Kapita[3]
Program Studi Informatika, Fakultas Teknik, Universitas Khairun
Jl. Jati Metro, Kota Ternate
E-mail*:* skymystery03@gmail.com[1], muhammadfhadli@unkhair.ac.id[2],
syarifuddinnkapita@unkhair.ac.id[3]

Penggunaan kata-kata gaul sering kali digunakan sebagai sarana komunikasi di media sosial seperti *Twitter*, namun menjadi masalah bagi kalangan tertentu karena sulit dimengerti jika diucapkan di luar konteks. Hal ini dapat menyebabkan komunikasi menjadi kurang efektif, terutama bagi yang tidak terbiasa dengan *slang* tersebut. Oleh karena itu, pendekatan normalisasi kata diperlukan untuk menerjemahkan kata ke dalam bahasa formal agar lebih dipahami masyarakat. *Natural Language Processing* (NLP) adalah teknik komputasi yang menganalisis dan merepresentasikan teks atau bahasa lisan untuk mencapai pemrosesan mirip manusia. Penelitian ini fokus pada teknik ekstraksi fitur seperti *FastText* dan *Word2Vec* untuk memetakan kata ke vektor numerik. Hasil pengujian kata *slang* menunjukkan *FastText* memiliki kemiripan tertinggi 0.9934859978 dan terendah 0.8928895496, sementara *Word2Vec* memiliki kemiripan tertinggi 0.9977979123 dan terendah 0.0975351095. Waktu yang dibutuhkan *FastText* untuk *training* adalah 0.432 detik dan untuk normalisasi 0.016 detik, sedangkan *Word2Vec* memerlukan 0.027 detik untuk *training* dan 0.006 detik untuk normalisasi.

Kata kunci: *Kata Slang, Natural Language Processing, Word2Vec, FastText*

BAB I

PENDAHULUAN

1.1. Latar Belakang

Kata *slang* merupakan istilah nonformal kerap digunakan oleh kalangan generasi muda, untuk menyampaikan ungkapan dan gagasan baik itu secara langsung maupun tidak langsung. *Slang* sering kali mencerminkan identitas generasi atau kelompok kultural tertentu. Berbagai kelompok usia atau komunitas dapat mengembangkan kata-kata atau frasa khusus yang hanya dimengerti oleh mereka. Ini dapat menimbulkan ketidakpahaman atas perbedaan pemahaman antar generasi (Samudro, 2019).

Menurut Samudro (2019) penggunaan kata *slang* sering kali digunakan sebagai sarana komunikasi antar pengguna internet di media sosial terutama *Twitter*. Namun kata *slang* ini menjadi sebuah permasalahan oleh kalangan tertentu. Salah satu *issue* yang muncul adalah bahwa kata *slang* dapat menjadi sulit dipahami jika diucapkan di luar konteks yang tepat. Ini dapat menciptakan situasi di mana komunikasi menjadi kurang efektif, terutama ketika orang yang tidak familiar dengan kata *slang*. Berdasarkan kesenjangan yang terjadi ada beberapa pendekatan yang digunakan pada penelitian ini, salah satunya pendekatan normalisasi.

Pendekatan normalisasi kata saat ini sangat diperlukan, guna mengartikan setiap kata ke dalam bahasa formal untuk memberikan pemahaman terhadap setiap kalangan masyarakat. Meskipun demikian, fenomena penggunaan kata-kata non baku ini dapat menjadi keuntungan bagi akademisi, banyak diantaranya yang memanfaatkannya sebagai bahan penelitian di bidang pemrosesan data teks. Data yang dihasilkan dari media sosial kemudian dapat diproses menggunakan metode yang dikenal sebagai *Natural Language*

Processing (NLP). *Natural Language Processing* (NLP) merupakan suatu pengembangan teknik komputasi bahasa alami dalam menganalisis dan merepresentasikan teks ataupun lisan untuk mencapai pemrosesan bahasa seperti bahasa manusia (Ramadhanti et al., 2019)

Terdapat banyak teknik maupun model yang dapat digunakan dalam NLP. Penelitian ini berfokus pada beberapa teknik ekstraksi fitur seperti *FastText* dan *Word2Vec*, yang mana dilakukan perbandingan pada setiap teknik untuk memetakan kata-kata ke dalam vektor numerik. Adapun perlu dilakukan tahapan-tahapan seperti pra-pemrosesan data, sebelum memulai proses pemrosesan data. Namun, dengan banyaknya variasi kata dalam bentuk tidak baku untuk suatu konteks yang sama, menjadi sulit dilakukan. Contohnya, kata 'semangat pagi' memiliki banyak variasi non baku seperti 'met pagi', 'mangats pagi', cemungut pagi' dan sebagainya. Tanpa melakukan pra-pemrosesan data, setiap kata tersebut dapat direpresentasikan sebagai kata yang berbeda.

Sejumlah penelitian sebelumnya telah dilakukan terkait klasifikasi data teks. Salah satunya penelitian (Nurdin et al., 2020) yang membandingkan kinerja *word embedding*, seperti *Word2Vec*, *GloVe*, dan *FastText*, yang diklasifikasikan menggunakan algoritma *Convolutional Neural Network*. Hasil penelitian tersebut menunjukkan nilai F-*measure* berturut-turut sebesar 0.925, 0.958, dan 0.979 (Nurdin et al., 2020). Penelitian lainnya membahas analisis sentimen dari *review* hotel dengan membandingkan akurasi model menggunakan *Word2Vec* dan *FastText*. Dalam penelitian ini, kedua metode *embedding* tersebut digabungkan dengan teknik *ensemble learning*, yakni *Random Forest*, *Extra Tree*, dan *AdaBoost*. Hasil terbaik diperoleh saat *FastText* digabungkan dengan *Random Forest* dan *Extra Tree*, masing-masing mencapai akurasi sebesar 93% (Khomsah et al., 2021).

Berdasarkan beberapa penelitian banyak literatur tentang analisis teks menggunakan

word embedding atau pengembangannya. Dari berbagai penelitian, sangat populernya penggunaan *word embedding* digunakan dalam bidang NLP, maka penelitian ini bertujuan untuk melakukan perbandingan kinerja dari model *word embedding* yaitu *FastText* dan *Word2Vec*. Penelitian ini akan membahas "Perbandingan Normalisasi Kata *Slang* Bahasa Indonesia Menggunakan Model *FastText* dan *Word2Vec* Dengan Pendekatan *Natural Language Processing*".

1.2. Rumusan Masalah

Berdasarkan latar belakang uraian masalah di atas, peneliti dapat merumuskan masalah pada penelitian ini yaitu bagaimana hasil perbandingan Model *FastText* dan *Word2Vec* dalam normalisasi dan memahami kata-kata *slang* Bahasa Indonesia.

1.3. Batasan Masalah

Berikut ini merupakan beberapa batasan masalah terkait dengan penelitian yang akan dilakukan:

1. Penelitian ini membandingkan dua model teknik NLP yaitu *Word2Vec* dan *FastText*.
2. Penelitian ini menggunakan data sumber dari media sosial yang terdiri dari teks dalam Bahasa Indonesia.
3. Penelitian ini nantinya melakukan teknik *scraping* dalam mengumpulkan data teks atau komentar dari media sosial.
4. Luaran dari penelitian ini merupakan dataset dan hanya berfokus pada normalisasi kata *slang* dan mengkonversi menjadi bentuk baku atau formal.
5. Bentuk implementasi pada penelitian ini hanya berupa model.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai pada penelitian ini adalah untuk mengetahui hasil dari perbandingan Model *Fastext* dan *Word2Vec* dalam normalisasi dan memahami kata-kata

slang Bahasa Indonesia.

1.5. Manfaat Penelitian

Penelitian ini diharapkan dapat bermanfaat sebagai berikut:

1. Penelitian ini dapat membantu dalam memahami lebih baik bahasa *slang* yang digunakan dalam komunikasi sehari-hari di masyarakat Indonesia.
2. Terkait dengan hasil dari perbandingan model NLP, penelitian ini dapat membantu mempelajari karakteristik maupun kinerja dari masing-masing model.
3. Penelitian ini berkontribusi dalam mengumpulkan data set khususnya pada Bahasa Indonesia yang telah dilakukan proses NLP hingga menjadi dataset yang siap digunakan untuk kepentingan pengembangan penelitian.

1.6. Sistematika Penulisan

BAB I PENDAHULUAN

Pada bab ini menjelaskan tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan untuk kasus yang akan dipecah.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan tentang teori-teori yang akan digunakan untuk penelitian Perbandingan Normalisasi Kata *Slang* Bahasa Indonesia Menggunakan Model *FastText* & *Word2Vec* Dengan Pendekatan *Natural Language Processing.*

BAB III METODE PENELITIAN

Bab ini menjelaskan langkah-langkah metode penelitian untuk Perbandingan Normalisasi Kata *Slang* Bahasa Indonesia Menggunakan Model *FastText* & *Word2Vec* Dengan Pendekatan *Natural Language Processing*.

BAB IV HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil penelitian yang telah dilakukan yang terdiri dari analisis data, pengujian metode, dan evaluasi metode yang menggunakan *Cosine Similarity*.

BAB V PENUTUP

Bab ini memuat kesimpulan dari hasil penelitian yang dilakukan dan saran untuk penelitian selanjutnya yang dapat dilakukan pengembangan mengenai topik terkait.

BAB II

TINJAUAN PUSTAKA

2.1. Penelitian Terkait

Penelitian terkait merupakan penelitian terdahulu yang berkaitan dengan penelitian yang sedang diteliti, untuk mengetahui perbedaan penelitian terkait dengan penelitian yang akan dilakukan dapat dilihat pada tabel 2.1.

Tabel 2. 1 Penelitian Terkait

No	Judul Penelitian	Metode Penelitian	Hasil Penelitian
1.	Membangun *Slang Dictionary* Untuk Normalisasi Teks Menggunakan *Pre-Trained FastText Model* (Lavenia Situmorang, Enjelin Hutahaean, Ruth Angeli Sibarani, Junita Amalia)	- Word Embedding dengan teknik Pre-Trained FastText - Algoritma Klasifikasi Deep Learning - Ekstraksi Fitur Continuous Bag Of Words(CBOW)	*Threshold value* yang digunakan untuk membangun *slang dictionary* adalah 0.05, 0.1, dan 0.2. Setelah didapatkan hasil dari *threshold value* yang digunakan untuk membangun *slang dictionary*, maka didapatkan hasil terbaik yaitu dari *threshold* dengan nilai 0.05 dikarenakan terdapat hasil *Accuracy* 0.65, *Precision* 1.00, *Recall* 1.00 dan *F-1 score* didapatkan hasil 1.00 (Situmorang et al., 2022).
2.	Perbandingan Kinerja *Word Embedding Word2Vec*, *Glove*, dan *FastText* Pada Klasifikasi Teks (Arliyanti Nurdin, Bernadus Anggo Seno Aji, Anugrayani Bustami, Zaenal Abidin)	- Pendekatan Natural Language Processing - Perbandingan 3 teknik Word Embedding (Word2Vec, Glove dan Fastext) - algoritma klasifikasi CNN - matriks evaluasi (Accuracy dan F-Masure)	Berdasarkan hasil eksperimen, kinerja CNN dalam mengklasifikasikan teks menggunakan *word embedding Word2Vec, GloVe,* dan *FastText* menggunakan ukuran *F-Measure* secara berturut-turut untuk *dataset* 20 *newsgroup* adalah 0.925, 0.958, dan 0.979, dan *dataset Reuters News* adalah 0.694, 0.688, dan 0.715. *Word2Vec* dan *GloVe* tidak mampu merepresentasikan vektor dari kata yang tidak ada dalam korpus (*out of vocabulary*). Berbeda dengan *FastText* yang dapat diandalkan untuk permasalahan *out of vocabulary*

			ini. Kinerja terbaik dari eksperimen diperoleh dengan menggunakan *word embedding FastText* (Nurdin et al., 2020).
3.	Perbandingan *Word* Embedding *Word2Vec*, Glove, dan *FastText* Menggunakan Deep Learning Pada Ulasan Kondisi Pengguna Obat Kesehatan (Fiqih Aulia Pradana)	- Perbandingan 3 Teknik Word Embedding (Word2Vec, GloVe dan FastText) - Algoritma Klasifikasi Deep Learning (LSTM) - Teknik Preprocessing Text (Missing Value, duplicate, case folding , stopwords dan lemmatization) - Teknik Evaluasi (Classification Rep ort)	Penggunaan *word* embedding *Word2Vec*, GloVe dan *FastText* dengan metode *deep learning* yaitu LSTM dalam melakukan klasifikasi terhadap data *Drugs Review* (ulasan kondisi pengguna obat kesehatan) mendapat hasil yang baik. Model *word* embedding *Word2Vec*, GloVe dan *FastText* dengan metode LSTM sangat baik dalam mengklasifikasikan kondisi pengguna obat kesehatan dari data *Drugs Review*, dan menghasilkan nilai akurasi berturut turut 85.20%, 84.19%, 86.22%. Model *word* embedding *FastText* dengan metode LSTM memiliki nilai akurasi dan F1-*Score* tertinggi dibandingkan kedua embedding lainnya dengan nilai akurasi 86.22% dan F1-*Score* 86% (Pradana, 2023).
4.	Normalisasi Teks Bahasa Indonesia Berbasis Kamus *Slang* Studi Kasus: Tweet Produk Gadget Pada *Twitter* (Riri Riyaddulloh & Ade Romadhony)	- Teknik Filtering (Pre-Processing) - Teknik Ekstraksi Fitur (TF-IDF) - Algoritma Klasifikasi Naive Bayes - Matrik Evaluasi (*Accuracy*)	Hasil akurasi yang diperoleh dalam pengujian dengan menggunakan variabel *Train* Data *Size* sebesar 80%, *Test* Data *Size* sebesar 20% diperoleh hasil akurasi dataset tanpa normalisasi sebesar 88% dan diperoleh hasil akurasi dataset dengan normalisasi menggunakan *Word2Vec* sebesar 91%, Sehingga dapat disimpulkan bahwa dataset setelah dilakukan proses normalisasi akan memiliki performansi klasifikasi yang lebih baik terhadap dataset yang belum dinormalisasi (Riyaddulloh & Romadhony, 2021).

5.	Normalisasi Teks Bahasa Indonesia Pada Media Sosial Berdasarkan *FastText Embeddings* (Ahmad Arif Samudro)	- Teknik Pengumpulan Data (Crawling) - Teknik *Filtering Pre Processing* - *Word Embedding FastText* dan *Word 2Vec*	Dari berbagai macam percobaan pembuatan model, ditemukan konfigurasi parameter *training* terbaik yang digunakan, yaitu modelV3_ft berbasis *FastText* dengan parameter *size*=300, *learning algorithm* CBOW, iterasi=2, *minimum word frequency*=5, *context window*=5, *min*_n=3, dan *max*_n=6. Akurasi yang didapatkan apabila diuji dengan 100 kata non-kamus adalah 45,6%. Dalam penelitian ini dapat disimpulkan bahwa model *FastText* sedikit lebih unggul dibandingkan model *Word2Vec*, dengan selisih akurasi pengujian 100 kata hanya sekitar 2~5% dengan parameter yang serupa (Samudro, 2019).

2.2. *Natural Language Processing*

Natural Language Processing atau pengolahan bahasa alami merupakan salah satu cabang ilmu yang memfokuskan penelitiannya pada pengembangan cara kerja komputer untuk memahami dan memproses bahasa alami dalam bentuk kalimat atau ucapan (Vig, 2019). Dalam NLP, tujuannya adalah untuk mengevaluasi bagaimana bahasa dalam suatu teks sesuai dengan aturan tata bahasa. Proses implementasi NLP melibatkan beberapa tahapan analisis bahasa, yaitu:

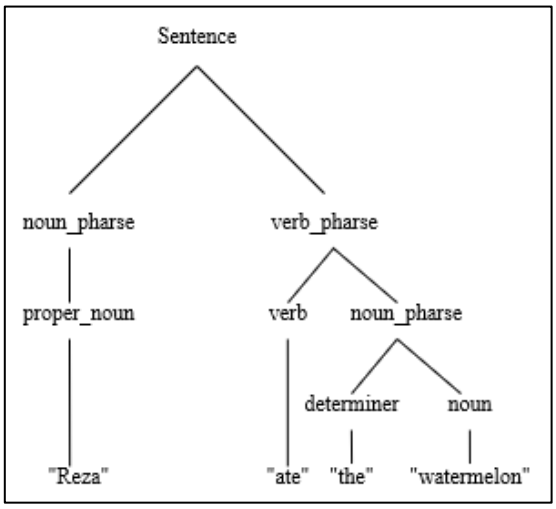
a. *Stemming* adalah proses pemotongan kata pada awal atau akhir kata untuk memperoleh kata dasar sesuai dengan aturan tata bahasa. Tujuan utama dari *stemming* ini adalah untuk menghilangkan imbuhan pada kata. Sebagai contoh, kata 'perumahan' akan disederhanakan menjadi 'rumah'.

b. *Tokenization*, merupakan proses pembagian sebagian kalimat menjadi kata per kata. Sebagai contoh, dalam kalimat "Rumah ini sebentar lagi akan ditinggalkan", setiap kata dalam kalimat tersebut dipisahkan satu per satu, yaitu "Rumah", "ini", "sebentar", "lagi", "akan", dan "ditinggalkan". Hasil dari pemisahan ini disebut sebagai token.

c. *Parsing*, adalah proses menentukan struktur pada teks dengan menganalisis kata penyusunnya berdasarkan tata bahasa yang mendasarinya. Contoh penerapannya dapat dilihat pada Gambar 2.1. Gambar 2.1 memberikan ilustrasi dari proses penguraian kata, di mana setelah proses tersebut, akan terbentuk sebuah *phrase tree* seperti yang ditunjukkan pada Gambar 2.2. Pada *phrase tree* ini, kalimat diuraikan kata per kata sesuai dengan tata bahasa yang berlaku.

Sentence > *noun_Pharse*, *verb_Pharse*
Noun_Pharse > *proper_noun*
Noun_Pharse > *determiner*, *noun*
Verb_Pharse > *verb*, *noun_Pharse*
Proper_noun > [Reza]
Noun> [*watermelon*]
Verb > [*ate*]
Determiner > [*the*]

Gambar 2. 1 Contoh Tata Bahasa



Gambar 2. 2 *Pharse Tree*

Berdasarkan Gambar 2.1, menunjukkan kalimat “Reza *ate the watermelon*”. Kalimat ini tentang seseorang yang sedang makan semangka. Secara tata bahasa, kalimat ini terdiri dari dua frasa yaitu frasa nominatif (*noun phrase*) dan frasa verbal (*verb phrase*). Frasa nominatif adalah subjek kalimat, yaitu “Reza”. Frasa verbal adalah predikat kalimat, yaitu

"*ate the watermelon*". Frasa nominal "Reza" adalah kata benda nama diri (*proper noun*). Frasa nominal "*the watermelon*" terdiri dari kata sandang (*determiner*) "*the*" dan kata benda "*watermelon*". Kata sandang "*the*" menunjukkan bahwa semangka yang dimaksud adalah semangka tertentu, yaitu semangka yang sedang dimakan oleh Reza. Frasa verbal "*ate the watermelon*" terdiri dari kata kerja (*verb*) "*ate*" dan frasa nominal "*the watermelon*". Kata kerja "*ate*" menunjukkan bahwa Reza melakukan Tindakan memakan semangka. Secara keseluruhan, kalimat "Reza *ate the watermelon*" adalah kalimat yang sederhana dan mudah dipahami. Kalimat ini menceritakan tentang tindakan yang dilakukan oleh Reza, yaitu memakan semangka.

Berdasarkan Gambar 2.2, menunjukkan diagram yang menjelaskan struktur kalimat dalam Bahasa Indonesia. Diagram tersebut dibagi menjadi dua bagian, yaitu bagian atas dan bagian bawah. Bagian atas diagram menunjukkan unsur – unsur yang membentuk kalimat, yaitu subjek, predikat, objek, dan keterangan. Subjek adalah unsur yang melakukan tindakan atau diterangkan dalam kalimat. Predikat adalah unsur yang menerangkan subjek. Objek adalah unsur yang dikenai tindakan oleh subjek. Keterangan adalah unsur yang memberikan informasi tambahan tentang subjek, predikat, atau objek. Bagian bawah diagram menunjukkan contoh kalimat yang menggunakan unsur-unsur tersebut. Contoh kalimat yang diberikan adalah "Reza makan semangka". Dalam kalimat tersebut, "Reza" adalah subjek, "makan" adalah predikat, "semangka" adalah objek, dan "di teras" adalah keterangan tempat. Berikut adalah penjelasan lebih rinci tentang masing-masing unsur kalimat:

a. Subjek

Subjek adalah unsur yang melakukan tindakan atau diterangkan dalam kalimat. Subjek dapat berupa kata benda (*noun*), frasa benda (*noun phrase*), atau klausa

(*clause*). Dalam contoh kalimat "Reza makan semangka", subjek adalah "Reza". "Reza" adalah kata benda nama diri (*proper noun*) yang merujuk pada orang tertentu.

b. Predikat

Predikat adalah unsur yang menerangkan subjek. Predikat dapat berupa kata kerja (*verb*), frasa kerja (*verb phrase*), atau klausa (*clause*). Dalam contoh kalimat "Reza makan semangka", predikat adalah "makan". "Makan" adalah kata kerja transitif yang memerlukan objek.

c. Objek

Objek adalah unsur yang dikenai tindakan oleh subjek. Objek dapat berupa kata benda (*noun*), frasa benda (*noun phrase*), atau klausa (*clause*). Dalam contoh kalimat "Reza makan semangka", objek adalah "semangka". "Semangka" adalah kata benda yang menunjukkan benda yang dimakan oleh Reza.

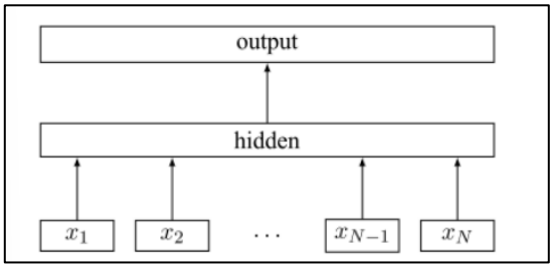
Menurut (Samudro, 2019)*Natural Language Processing* (NLP) adalah suatu proses komputasi yang digunakan pada komputer agar dapat menganalisa, memahami, dan memperoleh makna dari bahasa manusia yang digunakan dalam kehidupan sehari-hari. Proses komputasi ini diGambarkan sebagai suatu rangkaian kata yang memiliki aturan-aturan tertentu. Dalam melakukan NLP, terdapat beberapa kendala, kendala yang paling umum dialami adalah terjadinya ambiguitas, karena bahkan pada manusia, analisis makna dan konteks dalam suatu kalimat sangat dipengaruhi oleh pemahaman dan pengetahuan masing-masing individu. Kendala lain dalam proses ini adalah varian kosa kata yang semakin besar dan luas. Seiring berjalannya waktu, bahasa akan mengalami perkembangan, baik itu berupa penambahan ataupun perubahan. Model dari NLP akan didasarkan pada segi pengejaan kata, bagaimana kata-kata tersebut digabungkan menjadi satu kalimat, dan konteks kata yang ada pada kalimat tersebut. Dalam NLP terdapat

beberapa disiplin ilmu:

1. Fonetik/fonologi: Pengetahuan terkait suara yang membentuk kata yang bermakna.
2. Morfologi: Pengetahuan terkait bentuk-bentuk kata
3. Sintaksis: Pengetahuan terkait sintaksis/urutan kata dalam pembentukan kalimat.
4. Semantik: Pengetahuan terkait makna kata dan pembentukan arti kalimat dari kata-kata penyusunnya.
5. Pragmatik: Pengetahuan terkait konteks kata/kalimat yang berhubungan dalam suatu keadaan tertentu.

2.3. *FastText*

FastText adalah suatu model *word embedding* yang dikembangkan oleh tim riset AI *Facebook*. Model ini terinspirasi dari penelitian oleh Mikolov dkk. dan berhasil menunjukkan bahwa model mereka mampu melakukan *training* pada 1 milyar kata dalam waktu 10 menit, dengan kualitas hasil yang tidak kalah dengan model-model lainnya. Arsitektur model *FastText* mirip dengan arsitektur CBOW pada *Word2Vec* namun memiliki struktur hirarki dan merepresentasikan kata dalam bentuk *dense vector*. Dan diantara *input layer* dan *output layer* terdapat *hidden layer* seperti pada Gambar 2.3 (Samudro, 2019)



Gambar 2.3 Arsitektur *FastText*

Berdasarkan Gambar 2.3, *FastText* merupakan jenis *Word Embedding* yang dikembangkan oleh *Facebook*. *FastText* sendiri merupakan evolusi dari *Word2Vec*, yang sudah dikenal lebih lama sebagai representasi visual untuk *word Embedding*. *FastText*

memiliki beberapa keunggulan dibandingkan *Word2Vec*, salah satunya adalah kemampuannya dalam menangani kata-kata yang belum pernah dijumpai sebelumnya, yang dikenal sebagai *Out Of Vocabulary Word* (OOV). Sebagai contoh, kata non-baku seperti "Pengoptimalisasian" tetap dapat diberikan representasi vektor. Sebaliknya, pendekatan visual *Word2Vec* atau teknik *one-hot encoding* tradisional, dapat menghasilkan kesalahan ketika menghadapi kata yang tidak ada dalam kamus (Adam, 2019).

FastText memiliki kinerja yang baik, dapat melatih model pada dataset yang besar dengan cepat dan dapat memberikan representasi kata yang tidak muncul dalam data latih. Jika kata tidak muncul selama pelatihan model, kata tersebut dapat dipecah menjadi *n – gram* untuk mendapatkan *embedding* vektornya. Berikut adalah tahapan dalam perhitungan rumus *FastText*:

a. Pertama, membangun kamus kata – kata unik dari korpus teks yang digunakan.
b. Kedua, membangun sub – kata dari setiap kata dalam kamus. Sub – kata ini akan digunakan untuk memperhitungkan informasi morfologi dan struktur kata. Misalnya, kata ”makanan” dapat dipecah menjadi sub – kata ”makan” dan ”an”. Dalam *FastText*, sub – kata biasanya diwakili dengan menambahkan karakter khusus ”<” dan ”>” di awal dan akhir kata.
c. Ketiga, setiap kata dalam kamus akan diberikan *vector* awal yang diinisialisasi secara acak.
d. Keempat, hitung vektor konteks untuk setiap kata dalam kamus. Vektor konteks ini akan mencerminkan informasi sub – kata dari kata tersebut.
e. Kelima, menjumlahkan vektor representasi sub*word* n – grams berikut rumus 2.3.1.

$$v_{word} = v_{subword1} + v_{subword2} +, \dots, + v_{\text{sub}word\text{N}} \quad \text{..... (2.3.1)}$$

2.4. *Word2Vec*

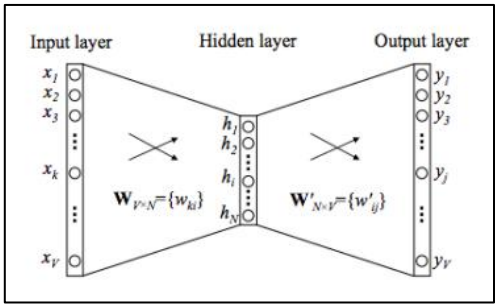
Word2Vec merupakan sekumpulan beberapa model yang saling berkaitan yang digunakan untuk menghasilkan *word embeddings*. *Word embeddings* merupakan sebutan dari seperangkat bahasa pemodelan dan teknik pembelajaran fitur pada *Natural Language Processing* (NLP) dimana setiap kata dari kosakata memiliki vektor yang mewakili makna dari kata tersebut dan kata-kata tersebut dipetakan ke dalam bentuk vektor bilangan riil. (Budiman et al., 2020) menjelaskan bahwa *Word2Vec* adalah dua lapisan *neural network* yang memproses teks. *Word2Vec* memiliki 2 algoritma belajar yaitu *Continuous BagOf-word* (CBOW) dan *Continuous Skip-Gram*. Dengan algoritma CBOW, urutan dari kalimat di dalam riwayat tidak mempengaruhi proyeksi. Algoritma ini memprediksi kata saat ini berdasarkan konteks. Sedangkan algoritma *skip-gram* memprediksi kata-kata yang berada disekitar suatu kata.

Word2Vec merupakan salah satu algoritma *word embedding* yang memetakan setiap kata dalam teks ke dalam vektor. Algoritma *Word2Vec* ini diciptakan oleh Mikolov dkk. pada tahun 2013. Sejak kemunculannya, model *word embedding* ini banyak digunakan dalam penelitian NLP. *Word2Vec* merepresentasikan kata ke dalam vektor yang dapat membawa makna semantik dari kata tersebut. Model *word embedding* ini merupakan salah satu aplikasi *unsupervised learning* menggunakan neural network yang terdiri dari sebuah *hidden layer* dan *fully connected layer*. Dimensi dari matriks bobot pada setiap layer adalah jumlah dengan kata dalam korpus dikalikan dengan jumlah *hidden* neuron pada *hidden layer*-nya. Matriks bobot pada *hidden layer* dari model yang telah dilatih digunakan untuk mentransformasikan kata ke dalam vektor. Matriks bobot ini seperti *lookup table*, di mana setiap baris mewakili setiap kata dan kolom mewakili vektor dari kata tersebut. *Word2Vec* mengandalkan informasi lokal dari bahasa. Semantik yang dipelajari dari kata tertentu dipengaruhi oleh kata-kata sekitarnya. Model ini mendemonstrasikan kemampuan untuk

mempelajari pola linguistik sebagai hubungan linear antar vektor kata. Terdapat dua algoritma *Word2Vec* yaitu *Continuous Bagof-word* (CBOW) dan *Skip-gram* (Nurdin et al., 2020)

a. CBOW

Model ini menggunakan konteks untuk memprediksi target kata. CBOW memiliki waktu *training* lebih cepat dan memiliki akurasi yang sedikit lebih baik untuk *frequent words*.



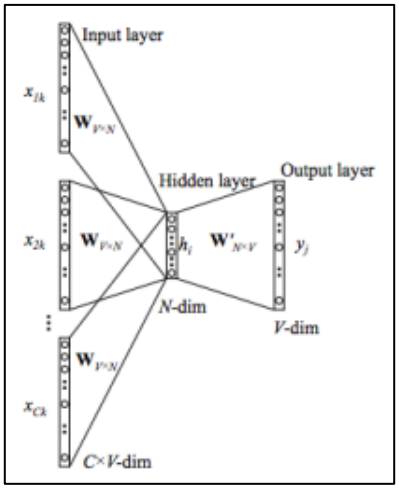
Gambar 2. 4 *One Word Context CBOW*

Berdasarkan Gambar 2.4, *one word context* CBOW adalah sebuah model pembelajaran mesin yang digunakan untuk mempelajari representasi vektor kata. Representasi vektor kata adalah representasi kata dalam bentuk vektor numerik yang dapat digunakan untuk melakukan berbagai macam tugas pengolahan bahasa alami (NLP), seperti klasifikasi teks, *clustering* teks, dan pencarian semantik. Dalam konteks *one word* con*text*, konteksnya hanya terdiri dari satu kata. Misalnya, pertimbangkan kalimat "Reza makan semangka". Dalam konteks *one word* con*text*, konteks kata "makan" adalah kata "Reza". Dengan demikian, model CBOW akan mencoba untuk memprediksi kata "makan" dari kata "Reza".

Lapisan *input* menerima kata-kata sebagai *input* dan menghasilkan representasi vektor numerik untuk setiap kata. Lapisan *hidden* melakukan transformasi pada

representasi vektor numerik dari lapisan *input* untuk mempelajari pola-pola yang kompleks dalam data. Lapisan *output* menghasilkan representasi vektor numerik akhir untuk setiap kata, yang dapat digunakan untuk melakukan berbagai macam tugas NLP.

Dalam konteks *one word* con*text*, lapisan *input* akan menerima kata "Reza" sebagai *input*. Lapisan *hidden* akan mempelajari pola-pola antara kata "Reza" dan kata "makan". Lapisan *output* akan menghasilkan representasi vektor numerik untuk kata "makan". Setelah model CBOW dilatih, representasi vektor numerik untuk kata "makan" dapat digunakan untuk melakukan berbagai macam tugas NLP. Misalnya, representasi vektor numerik ini dapat digunakan untuk mengklasifikasikan kalimat "Reza makan semangka" sebagai kalimat yang positif atau negatif.



Gambar 2. 5 *Multiple Context words CBOW*

Berdasarkan Gambar 2.5, *multiple context word* CBOW adalah sebuah model pembelajaran mesin yang digunakan untuk mempelajari representasi vektor kata. Dalam konteks *multiple context word*, konteksnya terdiri dari lebih dari satu kata. Misalnya, pertimbangkan kalimat "Reza makan semangka". Dalam konteks *multiple context word*, konteks kata "makan" adalah kata "Reza" dan kata "semangka".

Dengan demikian, model CBOW akan mencoba untuk memprediksi kata "makan" dari kata "Reza" dan kata "semangka".

Dalam konteks *multiple context word*, lapisan *input* akan menerima kata "Reza" dan kata "semangka" sebagai *input*. Lapisan *hidden* akan mempelajari pola-pola antara kata "Reza", kata "semangka", dan kata "makan". Lapisan *output* akan menghasilkan representasi vektor numerik untuk kata "makan".

Setelah model CBOW dilatih, representasi vektor numerik untuk kata "makan" dapat digunakan untuk melakukan berbagai macam tugas NLP. Misalnya, representasi vektor numerik ini dapat digunakan untuk mengklasifikasikan kalimat "Reza makan semangka" sebagai kalimat yang positif atau negatif.

b. *Skip – Gram*

Model ini menggunakan sebuah kata untuk memprediksi sasaran konteks. *Skip – gram* bekerja dengan baik menggunakan data pelatihan yang jumlahnya sedikit serta bisa merepresentasikan istilah-kata yang disebut langka. Arsitektur *Word2Vec Skip-gram* adalah kebalikan dari *Word2Vec* CBOW. Tujuan dari arsitektur *skip-gram* adalah untuk memprediksi konteks (*output*) disekitar kata tersebut (*input*). Adapun tahapan dari *Skip – gram*:

a. Pertama, melakukan pemrosesan pada teks yang akan digunakan

b. Kedua, membangun kamus kata yang berisi daftar kata unik yang ada dalam teks. Setiap kata dalam kamus akan diberi nomor indeks yang akan digunakan dalam perhitungan selanjutnya.

c. Ketiga, membangun pasangan kata yang akan digunakan dalam perhitungan *skip – gram*. Pasangan kata ini terdiri dari kata target dan kata konteks yang muncul dalam jarak tertentu di sekitar kata target.

d. Keempat, hitung probabilitas prediksi konteks kata – kata sekitarnya berikut rumus 2.1.

$$w_{t-n}, \dots, w_{t-1}, w_{t+1}, \dots w_{t+n} \quad \text{.....(2.1)}$$

Berdasarkan kata target W_t adalah seperti pada persamaan 2.2.

$$P(w_{t-n}, \dots, w_{t-1}, w_{t+1}, \dots w_{t+n} | w_t) = \prod_{c \in C} P(c|w_t) \quad \text{.....(2.2)}$$

Dengan, C : himpunan konteks kata – kata sekitarnya

Probabilitas dihitung menggunakan fungsi *softmax* seperti pada rumus 2.3.

$$P(w_{t+j}|w_t) = \frac{\exp\,(Vw_{t+j}.Vw_t)}{\sum w' \exp(Vw'.Vw_t)} \quad \text{.....(2.3)}$$

Dengan, P (Wt+j | Wt) : probabilitas bersyarat dari suatu peristiwa Wt+j terjadi pada waktu t+j diberikan bahwa peristiwa wt terjadi pada waktu t.

2.5. *Preprocessing*

Menurut (Riyaddulloh, 2021) *Preprocessing* merupakan langkah awal dalam proses *text mining* untuk menyiapkan teks menjadi data yang dapat diproses lebih lanjut. *Preprocessing* dilakukan berupa: huruf kecil (*lowercase*), *stemming* dan lain-lain agar kumpulan data tersebut dapat diproses pada proses selanjutnya. *Preprocessing* digunakan untuk membuat *dataset* sehingga dapat digunakan pada proses selanjutnya seperti pemilihan kata *slang*, normalisasi teks, dan seterusnya.

2.6. *Slang Dictionary*

Bahasa gaul, yang juga dikenal sebagai bahasa *slang*, merupakan variasi bahasa non-standar atau informal yang mengadopsi unsur bahasa prokem dari sebelumnya. Keberadaan bahasa *slang* muncul karena adanya suatu bentuk rahasia atau asosiasi dalam penggunaan bahasa. Ragam bahasa ini merupakan modifikasi dari berbagai bahasa yang tidak memiliki bentuk gaya bahasa yang jelas atau pasti. Ragam bahasa umumnya terdiri

dari singkatan, plesetan, dan terjemahan. Perbedaan kelompok kata yang digunakan dapat dilihat dari segi fonetik, leksikon, dan gramatika. Tujuan penggunaan bahasa gaul meliputi keperluan gurauan, untuk tampil berbeda dari yang lain, untuk menarik perhatian, menghindari kata-kata yang klise, mempersingkat kata, dan memudahkan dalam proses bersosialisasi. Bahasa gaul umumnya digunakan dalam situasi informal, dengan demikian, bahasa ini lebih sering dipergunakan oleh kalangan muda (Istiqomah et al., 2018).

2.7. *Twitter*

Twitter sebagai salah satu platform media sosial yang paling populer untuk komunikasi, memungkinkan pengguna untuk memposting pesan di situs pribadi mereka dengan batasan jumlah karakter tidak lebih dari 280. Meskipun tujuan awal *twitter* terbatas pada berbagi pesan antar pengguna, seiring berjalannya waktu, kemampuan baru ditambahkan, termasuk kemampuan untuk mengirim dan membaca pesan serta memposting pembaruan status. Platform ini juga dapat berfungsi sebagai wadah untuk perdagangan, sumber informasi, alat mobilisasi opini publik, dan sarana hiburan (Hannani, 2019).

2.8. *Google Colaboratory*

Google Colab atau *Google Colaboratory* adalah sebuah layanan *cloud computing* dari *Google* yang memungkinkan pengguna untuk menulis, mengeksekusi, dan berbagi kode *Python* secara gratis. Layanan ini memungkinkan pengguna untuk menulis dan mengedit program *Python* secara acak hanya dengan menggunakan *browser* web tanpa memerlukan konfigurasi. *Google Colab* juga memberikan akses gratis ke GPU dan dapat berbagi dengan mudah. *Notebook Colab* memungkinkan pengguna menggabungkan kode yang dapat dijalankan dalam satu dokumen, beserta Gambar, HTML, LateX, dan lainnya yang kemudian terintegrasi dan disimpan di akun *Google Drive* pengguna. Selain itu, pengguna dapat

memanfaatkan kecanggihan *library Python* yang popular untuk menganalisis dan memvisualisasikan data, juga dapat digunakan untuk mengimpor set data teks hingga Gambar, melatih pengklasifikasi, maupun mengevaluasi model (Colab, 2022).

2.9. *Python*

Bahasa pemrograman *Python* merupakan pilihan yang populer dan relatif mudah untuk dipelajari. *Python* sering digunakan dalam pengembangan perangkat lunak, kecerdasan buatan, pengembangan *web*, *machine learning*, dan analisis data. *Python* menyediakan berbagai *library*, seperti *NumPy* untuk komputasi numerik dan Pandas untuk analisis data, yang membantu mempermudah eksekusi tugas tertentu dengan cepat dan efisien (T I Sambi Ua et al., 2023).

2.10. *Cosine Similarity*

Metode algoritma cosine similarity sendiri adalah salah satu metode yang bisa dimanfaatkan sebagai metode pencarian data di dalam data mining dan sering digunakan untuk mendeteksi dokumen-dokumen yang mirip, cosine similarity akan menghitung tingkat kesamaan antar dua buah atau lebih dari objek yang dinyatakan dalam vektor jumlahnya ada dua vektor dengan menggunakan kata kunci (*cosine*). Jadi *cosine similarity* dapat digunakan untuk menemukan kemiripan dokumen dalam data set dengan jumlah yang besar dan dapat lebih cepat dan sesuai menemukan dokumen yang dicari (Kurniadi et al., 2020).

Cosine similarity atau kemiripan *cosinus* adalah ukuran jarak yang digunakan untuk data yang berupa vektor dokumen, yaitu dokumen yang dipandang sebagai data yang berisi ratusan atau bahkan ribuan atribut, dimana setiap atribut menyatakan sebuah *term* atau istilah (kata) yang nilainya berupa frekuensi kemunculan istilah dalam dokumen tersebut. Metode *cosine similarity* adalah metode yang digunakan untuk kesamaan atau kedekatan antar dokumen. Semakin besar nilai kesamaan vektor *query* dengan vektor dokumen maka

query tersebut dipandang semakin relevan dengan dokumen. Dengan demikian dua *vector cosinus* dari 1, dua *vector* pada 90˚ dengan orientasi yang sama memililki kesamaan 0. *Cosine similarity* terutama digunakan dalam ruang positif, dimana hasilnya dibatasi dengan (0,1). *Cosine similarity* kemudian memberikan tolak ukur seberapa mirip dua dokumen (Yusuf et al., 2018). Berikut adalah rumus *cosine similarity* seperti pada 2.4.

$$\cos(\theta) = \frac{A.B}{||A||.||B||} = \frac{\sum_{i=1}^{n}(A_I.B_i)}{\sqrt{\sum_{i=1}^{n} {A_i}^2}\sqrt{\sum_{i=1}^{n} {B_i}^2}} \quad \text{.....(2.4)}$$

Keterangan:

1. t = kata di *database*
2. d = dokumen
3. q = kata kunci/*query*
4. wij = bobot kata ke i pada dokumen j
5. wiq = bobot kata ke i pada dokumen

2.11. *Crawling* Data

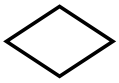
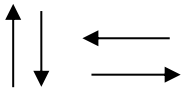
Crawling adalah istilah dalam teknologi informasi yang merujuk pada proses pengumpulan data dari internet secara otomatis dengan menggunakan perangkat lunak

khusus yang disebut web *crawler* atau *spider* (Akbar, 2023).

2.12. *Flowchart*

Flowchart adalah diagram yang menggambarkan alur atau urutan langkah-langkah dalam suatu program atau prosedur sistem secara logis. *Flowchart* atau yang disebut juga bagan alir, merupakan representasi grafis dari algoritma-algoritma yang terdapat dalam suatu program, yang mengindikasikan arah aliran dari program tersebut (Yulianeu, 2022). Selengkapnya dapat dilihat pada tabel 2.2.

Tabel 2. 2 *Flowchart*

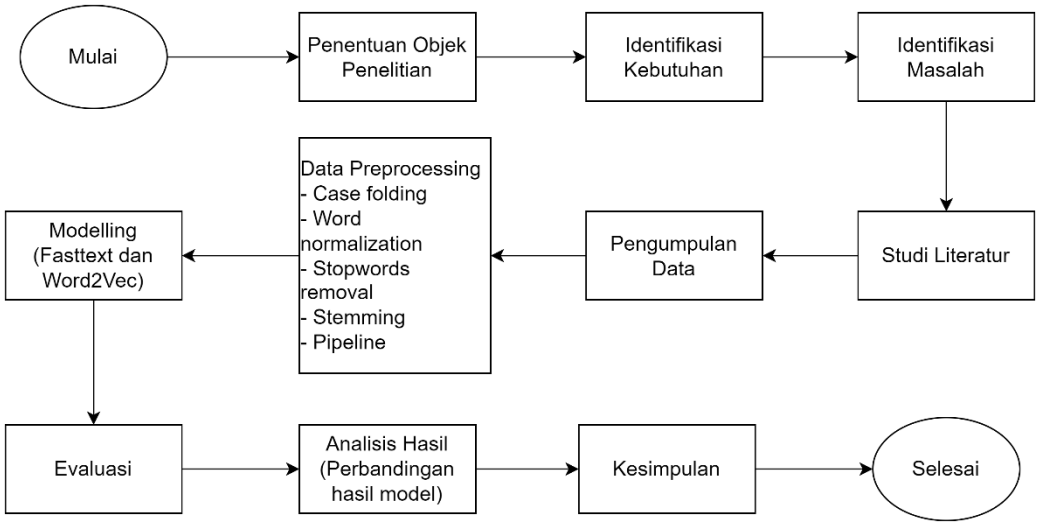
Simbol	Nama	Fungsi
	*Terminal*	Digunakan untuk memulai atau mengakhiri program.
	*Input/Output*	Digunakan untuk menyatakan input atau output tanpa melihat jenisnya.
	*Manual Operation*	Digunakan untuk menunjukkan pengolahan yang tidak dilakukan oleh komputer.
	*Decision*	Digunakan untuk memilih proses yang akan dilakukan berdasarkan kondisi tertentu.
	*Processing*	Digunakan untuk menunjukkan pengolahan data yang dilakukan oleh komputer.
	*Disk Storage*	Digunakan untuk menyatakan masukan dan keluaran yang berasal dari disk.
	*Flow Direction Symbol / Connecting Line*	Berfungsi untuk menghubungkan simbol yang satu dengan yang lainnya, menyatakan arus suatu proses.

BAB III

METODE PENELITIAN

3.1. Alur Penelitian

Tahapan penelitian dapat dilihat pada Gambar 3.1.



Gambar 3. 1 Alur Penelitian

Berikut ini adalah tahapan dalam penelitian ini untuk analisa perbandingan normalisasi kata *slang* Bahasa Indonesia menggunakan Model *FastText* dan *Word2Vec*.

3.2. Objek dan Waktu Penelitian

Objek yang digunakan dalam penelitian ini yaitu data kata *slang* Bahasa Indonesia pada media *twitter*. Waktu penelitian dilakukan pada semester ganjil tahun ajaran 2022/2023.

3.3. Alat Penelitian

Adapun alat dan bahan pada penelitian ini, terdiri dari perangkat keras (*hardware*) dan perangkat lunak (*software*) sebagai penunjang selama penelitian.

3.3.1. Perangkat Keras (Hardware)

Adapun spesifikasi perangkat keras (*hardware*) yang digunakan pada penelitian ini

sebagai penunjang proses perancangan dan pembuatan program selama penelitian berlangsung. Spesifikasi *hardware* dapat dilihat pada tabel 3.1.

Tabel 3. 1 Spesifikasi Perangkat Keras

Jenis	Spesifikasi
Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
RAM	8 GB DDR4
HDD	1 TB
SSD	222 GB
System Type	64-*bit operating system*, x64-*based processor*

3.3.2. Perangkat Lunak (*Software*)

Selain kebutuhan *hardware*, dibutuhkan juga *software* untuk melakukan perancangan dan pembuatan program. Spesifikasi perangkat lunak (*software*) dapat dilihat pada tabel 3.2.

Tabel 3. 2 Spesifikasi Perangkat Lunak

Nama	Keterangan
Windows 10 *Home*	Sistem operasi yang digunakan selama penelitian
Visual Studio Code	*Software* yang digunakan untuk menulis *script code.*

3.4. Identifikasi Masalah

Tahapan pertama yang dilakukan yaitu menganalisis masalah yang bertujuan untuk mengidentifikasi masalah yang ada mengenai normalisasi kata *slang* Bahasa Indonesia pada *twitter*.

3.5. Studi Literatur

Studi literatur merupakan tahapan mempelajari dan memahami teori – teori yang akan digunakan dalam penelitian. Studi literatur yang digunakan dalam penelitian ini yaitu mengumpulkan bahan referensi dari berbagai buku, jurnal, dan berbagai referensi lainnya.

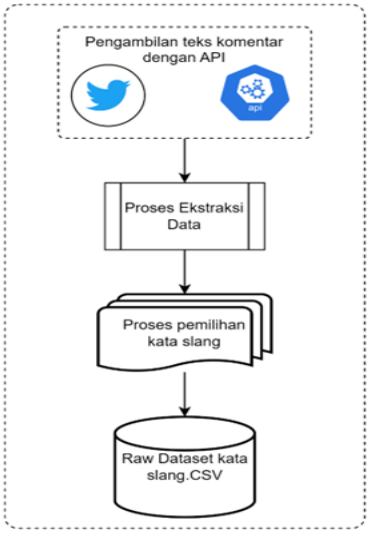
3.6. Pengumpulan Data

Tahapan keempat dalam penelitian ini penulis melakukan pengumpulan data yang merupakan bagian paling penting dalam penelitian. Data yang penulis kumpulkan

merupakan data sekunder yang diambil dari *twitter* yang terdiri dari 2500 data *tweet*. Tahapan pengumpulan data ini dilakukan dalam beberapa tahapan, yaitu:

3.6.1. Metode Pengumpulan Data

Pada tahapan ini merupakan sifat dari pendekatan induktif, dimana pada tahapan pengumpulan data, penelitian ini akan menggunakan teknik *scraping* (*crawling*), dimana teknik ini akan melakukan ekstraksi data, khususnya teks komentar pada media sosial. Berikut adalah Gambaran tahapan dari teknik *scraping*, dapat dilihat pada Gambar 3.2.



Gambar 3. 2 Proses Pengumpulan Data

Berikut merupakan penjelasan berdasarkan Gambar 3.2

1. Pengambilan Teks Komentar dengan API

   Proses pengambilan teks komentar dari media sosial (*twitter*) menggunakan API melibatkan beberapa tahapan. *Twitter* menyediakan API (*Application Programing Interface*) yang memungkinkan pengembang dan peneliti untuk mengakses data publik dari platformnya. Berikut ini merupakan penjelasan mengenai proses pengambilan teks komentar menggunakan API *Twitter*.

a. Pendaftaran Aplikasi

Pada tahap ini, perlu mendaftarkan akun *twitter* pada situs pengambangan *twitter* (https://developer.*twitter*.com). Ini melibatkan membuat akun pengembang, membuat proyek aplikasi, dan mendapatkan kunci API dan token akses yang diperlukan untuk mengakses data *twitter*.

b. Menentukan tujuan

Tahapan ini perlu di pertimbangkan guna menentukan tujuan terkait dengan teks komentar apa yang nantinya ingin digunakan

c. Menggunakan *EndPoint* API

Twitter memiliki berbagai *endpoint* yang memungkinkan dalam melakukan pengambilan data yang sesuai dengan kebutuhan penelitian. Salah satu *endpoint* yang sering digunakan adalah "*Search Tweets*" untuk mencari *tweet* yang sesuai dengan kata kunci atau parameter lainnya.

d. Autentikasi

Sebelum dapat mengakses data diperlukan *auth* token guna mendapatkan izin untuk mengakses *twitter*.

2. Proses Ekstraksi

Pada proses ini, dipelukan penggunaan bahasa pemrograman untuk melakukan pengambilan data. Pertama-tama, tahap *pre-processing* dimulai dengan proses *crawling* data dari *twitter* dengan fokus pada konteks produk gadget. Proses *crawling* dilaksanakan untuk mengumpulkan dataset yang akan digunakan dalam pembangunan model. Untuk melaksanakan *crawling* data *twitter*, diperlukan penggunaan API *twitter*, yang harus diajukan permohonannya kepada pihak *twitter*. Pengambilan data melalui *crawling* terbatas hingga maksimal 200 *tweet* per akun. Penelitian ini menggunakan bahasa pemrograman Python sebagai media

interpreternya dan *jupyter notebook* sebagai teks editor dalam menulis kode program. Pada tahapan ini, data teks nantinya akan di *crawling* sebanyak mungkin, guna mendapatkan *dataset* yang berukuran besar dan teks yang bervariasi.

3. Proses Pemilihan Kata *Slang*

   Pada proses ini, nantinya peneliti akan melakukan filterisasi bahasa yang terkandung pada data *raw*, guna memilah bahasa apa saja yang dipakai pada konteks penelitian ini.

4. *Raw Dataset* Kata *slang*

   Tahapan terakhir, dataset yang telah diproses pada tahapan-tahapan sebelumnya akan disimpan dalam bentuk file dengan ekstensi (CSV), untuk digunakan pada tahapan analisis data

3.7. Tahapan Data *Preprocessing*

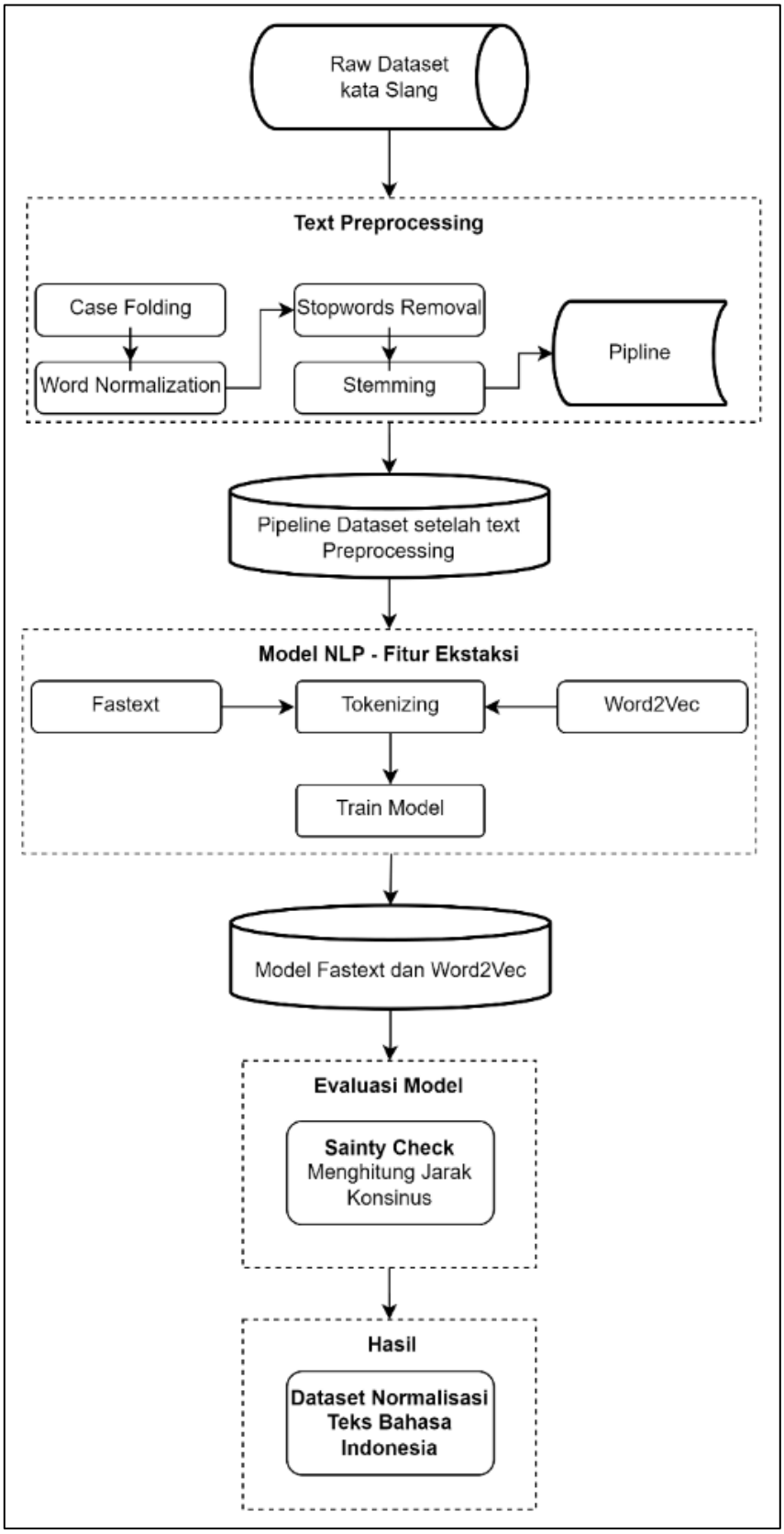
Pada tahapan ini, dilakukan proses metode kuantitatif berdasarkan *dataset* mentah yang telah dikumpulkan sebelumnya. Tujuannya adalah untuk melakukan evaluasi guna menjawab rumusan masalah yang telah ditetapkan. Untuk lebih jelasnya dapat dilihat pada Gambar 3.3. Berikut ini merupakan penjelasan dari proses analisis data:

1. *Raw Dataset* Kata *Slang*

   Dataset ini berisi teks yang masih dalam bentuk mentah dan akan diolah lebih lanjut dalam analisis selanjutnya.

2. *Text Preprocessing*

   Text Preprocessing adalah serangkaian langkah pra-pemrosesan yang digunakan dalam analisis teks dan pemrosesan bahasa alami untuk membersihkan dan mempersiapkan data teks sebelum analisis lebih lanjut. Beberapa komponen utama dalam teks *preprocessing* sebagai berikut pada gambar 3.3:



Gambar 3. 3 Proses Analisis Data

a. *Case Folding*

Case Folding adalah langkah mengubah semua karakter dalam teks menjadi huruf kecil (*lowercase*). Ini membantu menghindari masalah ketidak seragaman penulisan. Contohnya seperti penulisan “Kucing” dan “kucing” yang diperlakukan sebagai kata yang berbeda.

b. *Word Normalization*

Word Normalization adalah proses mengkonversi kata-kata ke bentuk dasarnya. ini dapat mencakup mengubah kata-kata bentuk tunggal seperti "kmu" menjadi “kamu”, atau mengganti kata-kata dengan sinonimnya.

c. *Stopwords Removal*

Stopwords adalah kata-kata umum seperti “dan”, “atau”, “di”, yang seringkali tidak memiliki nilai informasi atau makna dalam analisis teks. Kata-kata tersebut nantinya akan di eliminasi dari teks untuk mengurangi *noise* dan ukuran data.

d. *Stemming*

Stemming adalah proses menghapus awalan atau akhiran dari kata untuk mengembalikannya ke bentuk dasar (kata dasar). Contohnya “berjalan” dan “berjalanlah” menjadi jalan.

e. *Pipeline*

Pipeline adalah pendekatan sistematis yang menggabungkan langkah-langkah *preprocessing* ke dalam satu alur kerja yang berurutan. Ini memungkinkan untuk mengaplikasikan semua langkah *preprocessing* dengan mudah pada teks masukan.

3. *Pipeline Dataset* Setelah Teks *Preprocessing*

Pipeline teks *Preprocessing* membantu menjadikan data teks lebih bersih dan konsisten dengan cara menghilangkan karakter – karakter yang tidak perlu seperti

tanda baca, angka, simbol dan melakukan standarisasi format misalnya mengubah semua huruf menjadi *lowercase* sehingga memudahkan analisis selanjutnya. Ini juga membantu dalam meningkatkan kualitas hasil analisis teks.

4. Model NLP (Fitur Ekstraksi)

   Pada tahap ini, dilakukan pemodelan untuk setiap teknik, baik *Word2Vec* maupun *FastText*. Setiap model tersebut akan melalui tahap tokenisasi, di mana setiap kata akan dibuat dalam bentuk token, kemudian akan menjalani proses pelatihan untuk masing-masing model dalam sebuah berkas. Untuk model *FastText*, berkas tersebut akan memiliki ekstensi (.*FastText*), sementara untuk *Word2Vec*, berkas tersebut akan memiliki ekstensi (.w2v).

5. Evaluasi Model

   Pada tahap ini, akan dilakukan *sanity check* menggunakan perhitungan jarak cosinus untuk menguji setiap kata, terutama dalam pemahaman kata-kata yang memiliki makna yang serupa, pada masing-masing model, baik *FastText* maupun *Word2Vec*. Setelah itu, tahap ini akan mengevaluasi kelebihan dan kekurangan masing-masing model, terutama dalam fungsinya dalam memahami makna sebuah kata dalam teks.

6. Hasil

   Pada tahap ini, dataset telah melalui tahap normalisasi, di mana kata-kata *slang* yang awalnya digunakan telah diubah menjadi kata-kata dalam Bahasa Indonesia.

3.7.1. *Case Folding*

Case Folding adalah langkah mengubah semua karakter dalam teks menjadi huruf kecil (*lowercase*). Ini membantu menghindari masalah ketidak seragaman penulisan. Contohnya seperti penulisan “Kucing” dan “kucing” yang diperlakukan sebagai kata yang berbeda.

3.7.2. *Word Normalization*

Word Normalization adalah proses mengkonversi kata-kata ke bentuk dasarnya. ini dapat mencakup mengubah kata-kata bentuk tunggal seperti "kmu" menjadi “kamu”, atau mengganti kata-kata dengan sinonimnya.

3.7.3. *Stopwords Removal*

Stopwords adalah kata-kata umum seperti “dan”, “atau”, “di”, yang seringkali tidak memiliki nilai informasi atau makna dalam analisis teks. Kata-kata tersebut nantinya akan di eliminasi dari teks untuk mengurangi *noise* dan ukuran data.

3.7.4. *Stemming*

Stemming adalah proses menghapus awalan atau akhiran dari kata untuk mengembalikannya ke bentuk dasar (kata dasar). Contohnya “berjalan” dan “berjalanlah” menjadi jalan.

3.7.5. *Pipeline*

Pipeline adalah pendekatan sistematis yang menggabungkan langkah-langkah *Preprocessing* ke dalam satu alur kerja yang berurutan. Ini memungkinkan untuk mengaplikasikan semua langkah *preprocessing* dengan mudah pada teks masukan. *Pipeline* teks *preprocessing* membantu menjadikan data teks lebih bersih dan konsisten sehingga memudahkan analisis selanjutnya. Ini juga membantu dalam meningkatkan kualitas hasil analisis teks.

3.8. Proses Pemodelan (*Modelling*)

Pada tahap ini, dilakukan pemodelan untuk setiap teknik, baik *Word2Vec* maupun *FastText*. Setiap model tersebut akan melalui tahap tokenisasi, di mana setiap kata akan dibuat dalam bentuk token, kemudian akan menjalani proses pelatihan untuk masing-masing model dalam sebuah berkas. Untuk model *FastText*, berkas tersebut akan memiliki ekstensi

(.*FastText*), sementara untuk *Word2Vec*, berkas tersebut akan memiliki ekstensi (.w2v).

3.9. Evaluasi

Pada tahap ini, akan dilakukan *sanity check* menggunakan perhitungan jarak cosinus untuk menguji setiap kata, terutama dalam pemahaman kata-kata yang memiliki makna yang serupa, pada masing-masing model, baik *FastText* maupun *Word2Vec*. Setelah itu, tahap ini akan mengevaluasi kelebihan dan kekurangan masing-masing model, terutama dalam fungsinya dalam memahami makna sebuah kata dalam teks.

3.10. Analisis Hasil

Pada tahap ini, *dataset* telah melalui tahap normalisasi, di mana kata-kata *slang* yang awalnya digunakan telah diubah menjadi kata-kata dalam Bahasa Indonesia.

3.11. Kesimpulan

Pada tahap kesimpulan pada penelitian ini merupakan tahap akhir dari penelitian yaitu penarikan kesimpulan dari hasil analisis data yang telah dilakukan sebelumnya untuk menjawab rumusan masalah.

3.12. Praproses Teks

Data yang diperoleh melalui proses *crawling* sebelumnya akan mengalami tahap *cleansing* guna mengeliminasi unsur kata yang tidak diperlukan. Data *tweet* yang telah dihasilkan dari proses *crawl* masih memerlukan proses *preprocessing* untuk memudahkan penggunaan data dalam pembangunan model normalisasi teks. *Preprocessing* merupakan langkah awal dalam pengolahan data, dimulai dari data non-struktural hingga menjadi data terstruktur, dengan tujuan mencapai tingkat relevansi yang optimal antara data, dokumen, atau kata dengan kategori yang ditetapkan. Proses *preprocessing* yang dilakukan dalam Tugas Akhir ini melibatkan langkah-langkah *cleansing*. *Cleansing* adalah proses membersihkan kata-kata yang tidak diperlukan guna mengurangi *noise*. Tahapan dalam

cleansing melibatkan:

a. *Case Folding*

Tabel 3. 3 Contoh Penerapan *Case Folding*

Teks	\gue suka banget naik gunung, cuman gue Gak PUNYA KAKI BANG457!! 😭 https://t.co/LqK2374f01
Case Folding	gue suka banget naik gunung cuman gue gak punya kaki bang

Pada tabel 3.3 , merupakan tahapan *case folding*, tahapan ini semua huruf 'a' sampai 'z' pada dokumen teks diubah menjadi huruf kecil (*lowercase*) selain itu dihapus atau diabaikan dianggap sebagai *delimiter*. *Case folding* berperan untuk menyamaratakan penggunaan huruf kapital karena dua kata yang sama misalnya "*Twitter*" dan "*Twitter*" dianggap berbeda karena memiliki struktur huruf yang berbeda.

b. *Word Normalization*

Tabel 3. 4 Contoh Penerapan *Word Normalization*

Teks	GUE NGERTI KALO ELU ITU SUKA MAKAN BAKSO OTAK MANUSIA
word normalization	gue ngerti kalo elu itu suka makang bakso otak manusia

Pada tabel 3.4, merupakan tahapan *word normalization*, *word normalization* adalah proses mengubah bentuk-bentuk kata ke dalam bentuk standar agar lebih mudah diproses dan dianalisis. Contoh strategi *word normalization* melibatkan penggunaan *stemming*, *lemmatization*, dan penghapusan tanda baca untuk menciptakan konsistensi dalam representasi kata-kata dalam teks.

c. *Stopword Removal*

Tabel 3. 5 Contoh Penerapan *Stopwords Removal*

Teks	menaiki kadang gue suka banget naik gunung, cuman gue Gak PUNYA KAKI BANG457!! 😭 https://t.co/LqK2374f01

Stopwords Removal	kadang suka gunung cuman gak punya bang457 https t co lqk2374f01

Pada tabel 3.5, merupakan tahapan *stopwords removal*, tahapan menghilangkan kosakata yang bukan merupakan kata unik, kata yang tidak relevan, dan kata yang tidak bermakna. Misalnya, penggunaan kata penghubung seperti dan, yang, serta, setelah, dan lainnya. Dengan melakukan *stopwords removal* ini dapat mengurangi *noise* serta mempercepat waktu pemrosesan.

d. *Stemming*

Tabel 3. 6 Contoh Penerapan *Stemming*

Teks	menaiki kadang gue suka banget naik gunung, cuman gue Gak PUNYA KAKI BANG457!! 😭 https://t.co/LqK2374f01
stemming	kadang suka gunung cuman gak punya bang https t co lqk2374f01

Pada tabel 3.6, merupakan tahapan *stemming*, proses *stemming* dilakukan menggunakan *library* Sastrawi dengan tujuan untuk menghasilkan bentuk kata yang seragam. Sebagai contoh, kata "menyesal" setelah menjalani proses *stemming* dengan Sastrawi akan berubah menjadi kata dasarnya, yaitu "sesal". Beberapa tahapan yang dilakukan oleh *library* Sastrawi melibatkan pemeriksaan apakah kata tersebut merupakan akar kata atau tidak, penghilangan *prefix*, *suffix*, dan *infix*. Jika tahapan-tahapan tersebut tidak dapat diselesaikan atau mengalami kegagalan dalam pengubahan, maka kata tersebut akan dikembalikan ke bentuk aslinya.

e. *Pipeline*

Pipeline adalah serangkaian tahapan atau proses terstruktur yang diterapkan secara berurutan dalam pemrosesan bahasa alami untuk mencapai tujuan analisis teks. Setiap tahap dalam *pipeline* memiliki tanggung jawab spesifik dalam mengolah dan

mengubah data teks.

f. *Tokenizing FastText*

Tokenizing dalam *fasttext* melibatkan langkah-langkah serupa dengan *tokenizing* dalam *word2vec*, dengan penekanan pada pengelompokan kata menjadi subkata atau *n-grams*, yang dapat meningkatkan pemahaman representasi kata, terutama untuk kata-kata yang tidak umum atau kata-kata yang tidak ada dalam korpus pelatihan.

g. *Train Model FastText*

Pelatihan model *fasttext* melibatkan beberapa langkah yang mirip dengan pelatihan model *word2vec*. Namun, *fasttext* memiliki keunggulan tambahan karena mampu menangani kata-kata yang tidak umum atau tidak terlihat dalam korpus pelatihan melalui penggunaan subkata atau *n-grams.*

Cara Kerja *FastText*

Contoh kata : "*apple*"

Pertama, pembentukan *subword n – grams*:

Subword n – grams: "app", "ppl", "ple", "le"

Kedua, pembentukan *vector* representasi kata:

$V_{"app"}$ = [0.2, 0.4, 0.6]

$V_{"ppl"}$ = [0.1, 0.3, 0.5]

$V_{"ple"}$ = [0.3, 0.5, 0.7]

$V_{"le"}$ = [0.4, 0.6, 0.8]

Vektor representasi kata "*apple*" (V_w) diperoleh dengan menjumlahkan *vector* representasi *subword n – grams* yang terkandung dalam kata "*apple*":

$$v_{word} = v_{"app"} + v_{"ppl"} + v_{"ple"} + v_{"le"} \quad \text{..... (2.3.1)}$$

$= [0.2, 0.4, 0.6] + [0.1, 0.3, 0.5] + [0.3, 0.5, 0.7] + [0.4, 0.6, 0.8]$

$= [1.0, 1.8, 2.6]$

h. *Tokenizing Word2Vec*

Tokenizing dalam konteks *word2vec* merujuk pada pemisahan teks menjadi unit-unit dasar yang disebut token, yang biasanya adalah kata-kata. *Tokenizing* menjadi langkah awal yang penting dalam persiapan data untuk melatih model *word2vec*.

i. *Train Model Word2Vec*

Melatih model *word2vec* melibatkan dua pendekatan utama yaitu *Skip-gram* dan *Continuous Bag of words* (CBOW). Kedua metode ini menggunakan arsitektur jaringan saraf tiruan (*neural network*) untuk mempelajari vektor representasi kata yang mencerminkan hubungan semantik antar kata dalam korpus teks.

Cara Kerja *Word2Vec*

Contoh kalimat:

“Saya suka makan nasi goreng.”

Pertama, membagi kalimat menjadi kata – kata dan membangun kamus kata. Dalam contoh ini, kamus kata kita terdiri dari kata – kata berikut: ”saya”, ”suka”, ”makan”, ”nasi”, ”goreng”.

Kedua, misalkan kita menggunakan dimensi vektor 3 untuk representasi kata – kata. Hasilnya bisa seperti ini:

Vektor representasi kata “saya” = [0.2, 0.4, 0.6]

Vektor representasi kata “suka” = [0.1, 0.3, 0.5]

Vektor repre78sentasi kata “makan” = [0.3, 0.6, 0.9]

Vektor representasi kata “nasi” = [0.4, 0.8, 1.2]

Vektor representasi kata “goreng” = [0.5, 1.0, 1.5]

Misalkan kita menggunakan jendela konteks dengan jarak n = 1. Dalam hal ini, konteks kata – kata sekitarnya untuk kata target ”makan” adalah ”suka” dan ”nasi”.

Ketiga, menghitung probabilitas prediksi konteks kata – kata ”suka” dan ”nasi” berdasarkan kata target ”makan” menggunakan rumus:

$$Kalimat = \sum_{i=1}^{n} Vektor_i$$

$$Kalimat = Vektor_1 + Vektor_2 + Vektor_3 + Vektor_4 + Vektor_5$$

$$Vektor_{Kalimat} = [0.2, 0.4, 0.6] + [0.1, 0.3, 0.5] + [0.3, 0.6, 0.9] + [0.4, 0.8, 1.2] + [0.5, 1.0, 1.5]$$

$$Vektor_{Kalimat} = [0.2 + 0.1 + 0.3 + 0.4 + 0.5,\ 0.4 + 0.3 + 0.6 + 0.8 + 1.0,\ 0.6 + 0.5 + 0.9 + 1.2 + 1.5]$$

$$Vektor_{Kalimat} = [1.5, 3.1, 4.7]$$

Sebagai contoh, vektor representasi kata "saya" adalah [0.2, 0.4, 0.6], "suka" adalah [0.1, 0.3, 0.5], "makan" adalah [0.3, 0.6, 0.9], "nasi" adalah [0.4, 0.8, 1.2], dan "goreng" adalah [0.5, 1.0, 1.5]. Dengan menggunakan jendela konteks dengan jarak n = 1, konteks kata-kata sekitar kata target "makan" adalah "suka" dan "nasi".

Untuk menghitung vektor representasi kalimat, dilakukan penjumlahan vektor-vektor kata penyusunnya. Vektor kalimat diperoleh dengan menjumlahkan semua vektor kata, yaitu [0.2, 0.4, 0.6] + [0.1, 0.3, 0.5] + [0.3, 0.6, 0.9] + [0.4, 0.8, 1.2] + [0.5, 1.0, 1.5]. Dengan demikian, kalimat "Saya suka makan nasi goreng" dapat direpresentasikan dalam bentuk vektor [1.5, 3.1, 4.7].

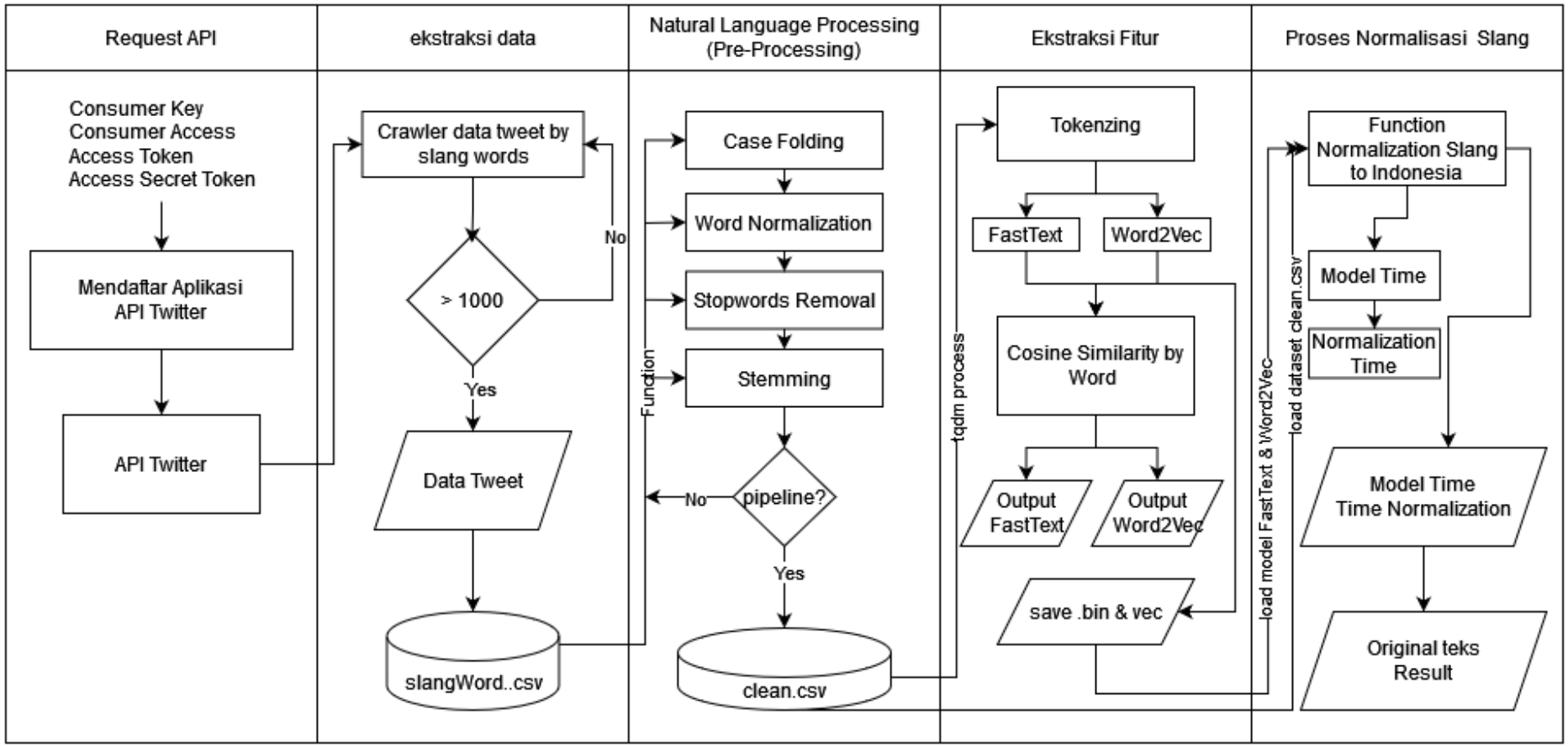
BAB IV

HASIL DAN PEMBAHASAN

Pada penelitian ini, pengumpulan data dilakukan dengan menggunakan teknik crawling dari media sosial Twitter. Crawling merupakan proses mengambil atau mengumpulkan data secara otomatis dari sumber data seperti halaman web, media sosial, atau platform lainnya. Proses crawling pada Twitter biasanya dilakukan dengan memanfaatkan API (Application Programming Interface) yang disediakan oleh platform tersebut. API memungkinkan pengembang untuk mengakses dan mengambil data terprogram melalui kode atau skrip yang ditulis.

Dalam penelitian ini, sebanyak 1000 data tweet berhasil dikumpulkan melalui proses crawling. Pengumpulan data dilakukan dengan membuat program atau skrip yang terhubung dengan API Twitter. Program ini kemudian mengambil data tweet sesuai dengan kriteria yang diinginkan, seperti kata kunci tertentu, rentang waktu, lokasi, atau bahkan tanpa mengandung kata tertentu. Data tweet yang terkumpul kemudian disimpan dalam format yang sesuai untuk diproses lebih lanjut, seperti file teks atau basis data.

Teknik crawling dari media sosial seperti Twitter memungkinkan pengumpulan data dalam jumlah besar secara efisien dan terstruktur. Data yang diperoleh dapat digunakan untuk berbagai tujuan, seperti analisis sentimen, pemantauan tren, atau pengembangan model machine learning untuk pemrosesan bahasa alami. Untuk lebih jelas dapat dilihat pada gambar 4.1.

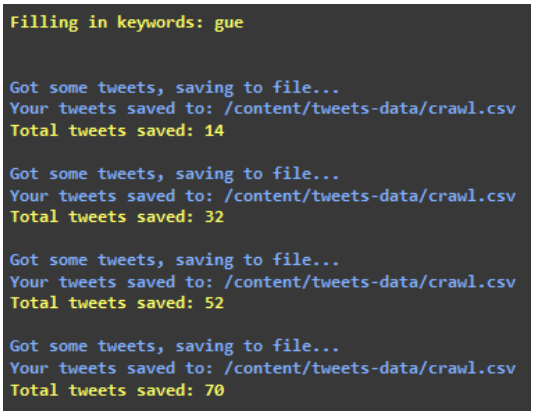


Gambar 4. 1 *Workflow Modelling Natural Language Processing*

4.1. *Request* API

Proses pengambilan teks komentar dari media sosial *Twitter* menggunakan API melibatkan beberapa tahapan. *Twitter* menyediakan *Application Programming Interface* (API) yang memungkinkan pengembang dan peneliti untuk mengakses data publik dari platform tersebut. Untuk dapat melakukan analisis data *Twitter* yang komprehensif, diperlukan akses ke data mentah dari platform tersebut dalam jumlah besar. Tujuan penelitian ini adalah mengumpulkan data *Twitter* menggunakan alat otomatisasi sumber terbuka bernama *tweet-harvest*. Alat ini dibangun dengan bahasa pemrograman Node.js, sehingga pertama-tama perlu mengonfigurasi lingkungan penelitian dengan menginstal platform tersebut. Setelah Node.js berhasil terkonfigurasi, juga diimpor pustaka Python 'pandas' yang akan digunakan nantinya untuk memanipulasi dan menganalisis dataset *Twitter* dalam format tabular. Data *Twitter* mentah hasil koleksi *tweet-harvest* selanjutnya akan diolah menggunakan teknik pembelajaran mesin untuk mengidentifikasi berbagai pola dan tren topikal dari cuitan pengguna.

4.2. Ekstraksi Data



Gambar 4. 2 *Output* Proses *Crawling* Data

Berdasarkan gambar 4.2 tersebut, tetelah lingkungan pengembangan terkonfigurasi langkah selanjutnya adalah mendefinisikan parameter-parameter untuk proses *crawling*

data dari *Twitter*. Pertama, ditentukan nama file *output* CSV tempat data yang di-*crawl* akan disimpan, yaitu *crawl*.csv. Kemudian ditentukan kata kunci pencarian yang akan digunakan oleh *tweet-harvest*, yaitu "gue". Selanjutnya, *limit* pencarian ditetapkan sebesar 1000 cuitan. Ini adalah jumlah maksimum data yang akan diambil oleh *tweet-harvest* dari *Twitter* berdasarkan kata kunci yang ditentukan sebelumnya. Parameter otentikasi token *Twitter* juga perlu ditambahkan agar *tweet-harvest* dapat mengakses API *Twitter*. Setelah semua parameter didefinisikan, perintah *tweet-harvest* kemudian dieksekusi untuk mulai mengumpulkan data. Proses *crawling* diperkirakan akan memakan waktu beberapa menit hingga *limit* 1000 cuitan tercapai. File *output crawl*.csv akan berisi data mentah cuitan-cuitan pengguna *Twitter* yang mengandung kata kunci "gue".

4.3. Natural Language Processing (NLP)

Berdasarkan hasil *import* data csv dengan pandas, telah dilakukan percobaan membaca file 'penelitian_rifqah.csv' dengan beberapa *encoding* yang berbeda yaitu 'utf-8', 'latin1', 'ISO-8859-1', dan 'cp1252'. Setiap *encoding* dicoba satu per satu dalam perulangan *for*. Jika *encoding* berhasil membaca file csv tanpa *error*, maka program akan mencetak *encoding* yang digunakan dan menampilkan 20 baris pertama dari data yang berhasil dibaca. Kemudian perulangan akan dihentikan dengan perintah *break*. Jika *encoding* gagal membaca data karena *Unicode Decode Error*, maka program akan mencetak pesan bahwa *encoding* tersebut gagal digunakan. Percobaan *encoding* selanjutnya akan dilakukan pada iterasi berikutnya dalam perulangan. Dengan strategi ini file csv dapat dibaca dengan mencoba beberapa kemungkinan *encoding* secara otomatis sampai ditemukan *encoding* yang cocok untuk membaca file csv tertentu. Hasil akhirnya adalah *encoding* yang berhasil digunakan untuk membaca file csv beserta data yang berhasil di-*parsing* ke dalam *dataframe* Pandas.

```
File read successfully with encoding: utf-8
                                                full_text
0    @sunmoonxzy sangat di tunggu kelanjuttannya  t...
1                                                 Njier
2     @tanyarlfes Rutinitas anak kos yg mager ke dapur
3    Yomi klo gak ada aku bebersih rumah mulu  klo ...
4    the real di rumah full seharian bener bener ma...
..                                                 ...
976  Kabarnya  dia terlibat dalam pembuatan video a...
977  Namun  dia sendiri membantah kabar tersebut da...
978  Gak cuma itu  bro  masih banyak lagi gosip art...
979  Ada gosip tentang si HM  aktris yang dikabarka...
980  Kabarnya  mereka sering bertemu di tempat-temp...

[981 rows x 1 columns]
```

Gambar 4. 3 *Output Dataset*

Dari gambar 4.3 *output* program tersebut, dapat dianalisis bahwa program tersebut berhasil melakukan pembacaan file dengan menggunakan pengodean utf-8. Teks yang ditampilkan pada *output* program terdiri dari beberapa baris kalimat yang merupakan hasil dari proses *crawling* data dari *Twitter*.

4.3.1. *Case Folding*

```python
import pandas as pd
import re

pola = r'@\w+'
pola_https = r'https:?://\S+|www\.\S+'
pola_spasi = r'\s+'
pola_tanda_kurung = r'\.\.'
pola_tanda_pagar = r'\(|\)'
pola_pagar = r'#'
pola_angka = r'\d+'
pola_karater = r'[!%.,]'
emotikon = r'[^\x00-\x7F]+'
hapus_karakter = r'[?_]'
karakter_siku = r'[^\w\s,:;+/^~@""*&]'

def casefolding(text):
    text = re.sub(pola, '', text)
    text = re.sub(pola_https, '', text)
    text = re.sub(pola_spasi, ' ', text)
    text = re.sub(pola_tanda_kurung, '', text)
    text = re.sub(pola_tanda_pagar, '', text)
    text = re.sub(pola_pagar, '', text)
    text = re.sub(pola_angka, '', text)
    text = re.sub(pola_karater, '', text)
    text = re.sub(emotikon, '', text)
    text = re.sub(hapus_karakter, '', text)
    text = re.sub(karakter_siku, '', text)
    return text.lower()

text = "@Komentar_Tukiyem \gue suka banget naik gunung, cuman gue Gak PUNYA KAKI BANG457!! https://t.co/LqK2374f01"
print("Input: {}".format(text))
print("Output: {}".format(casefolding(text)))
```

Gambar 4. 4 *Case Folding* dalam Program

Berdasarkan gambar 4.4, pada tahapan ini dilakukan *preprocessing text* dengan mendefinisikan beberapa pola *regex* untuk menghilangkan berbagai jenis *noise*. Pola-pola

regex tersebut digunakan untuk menghapus tanda '@', url/link, spasi berlebih, tanda kurung, tanda pagar, angka, tanda baca, karakter emoji, karakter *underscore*, dan karakter kurung siku. Seluruh pola *regex* digabungkan dalam fungsi *preprocess_text* untuk memproses satu *string text*. Dalam fungsi tersebut, masing-masing pola di-*match* dan dihapus dari *text input* menggunakan *method sub* () dari *regex*. *Text* yang telah dibersihkan kemudian dikonversi menjadi huruf kecil dengan *lower* (). Contoh kasus *text* berisi *mention*, link, angka, emoji, dan tanda baca telah berhasil dibersihkan. Karakter-karakter tersebut terhapus dari *text* dan *output* yang dihasilkan hanyalah rangkaian kata-kata bersih tanpa *noise*. Fungsi *preprocess_text* ini kemudian diterapkan untuk setiap teks pada kolom tertentu dalam *dataframe* menggunakan *method apply* (). Sehingga keseluruhan data menjadi lebih terstruktur dan siap untuk analisis *text* selanjutnya.

Tabel 4. 1 Contoh *Case Folding* Pada Dataset

Original Text	**Case Folding**
@Komentar_Tukiyem \gue suka banget naik gunung, cuman gue Gak PUNYA KAKI BANG457!! 😭 https://t.co/LqK2374f01	gue suka banget naik gunung cuman gue gak punya kaki bang

Tabel 4.1 diatas merupakan *output* dari contoh *case folding* kalimat *input* tersebut berhasil ditransformasikan menjadi *output* yang hanya berisi kata-kata penting dalam bentuk kata dasar. *Output* akhir yang dihasilkan adalah "gue suka banget naik gunung cuman gue gak punya kaki bang", yang merupakan representasi bersih dari kalimat *input* setelah diproses dengan algoritma pengolahan teks yang tepat.

4.3.2. *Word Normalization*

Pada tahapan ini dilakukan *Import* kamus *slang* ke dalam variabel *slang_dict* dengan membaca file csv '*slangdict.csv*' menggunakan fungsi *read_csv()* dari pandas. Kamus *slang* ini berupa *dataframe* yang berisi pasangan kata *slang* dan maknanya dalam bahasa baku.

```
def textnormalize(text):
    words = text.split()
    normalized_words = []

    for word in words:
        matching_rows = key_norm[key_norm['slang'] == word]
        if not matching_rows.empty:
            normalized_word = matching_rows['dictionary'].values[0]
            if pd.isna(normalized_word):
                normalized_word = word
        else:
            normalized_word = word
        normalized_words.append(str(normalized_word))

    normalized_text = ' '.join(normalized_words)
    normalized_text = normalized_text.lower()

    return normalized_text

text = "Yomi klo gak ada aku bebersih rumah mulu"
print("Input: {}".format(text))
print("Output: {}".format(textnormalize(text)))
```

✓ 0.0s

Gambar 4. 5 *Word Normalization* dalam Program

Berdasarkan gambar 4.5, kamus *slang* ini nantinya digunakan untuk melakukan *text normalization*, yaitu mengganti kata-kata tidak baku dan singkatan yang terdapat dalam teks menjadi bentuk baku. Dengan adanya kamus *slang*, *text* yang mengandung kata tidak baku dapat ditranslasikan ke dalam bahasa baku sehingga lebih mudah dipahami dan diproses untuk kebutuhan analisis teks. Tahapan *text normalization* ini merupakan bagian dari *preprocessing* teks agar menjadi lebih terstruktur.

Tabel 4. 2 Contoh *Word Normalization* Pada Dataset

Original Text	**Word Normalization**
Yomi klo gak ada aku bebersih rumah mulu	yomi kalau tidak ada aku bersih rumah melulu

Pada tabel 4.2 diatas, ditampilkan *input* berupa sebuah kalimat "Yomi klo gak ada aku bebersih rumah mulu" yang merupakan kalimat tidak baku dengan penggunaan kata *slang* seperti "gak" dan "bebersih". Setelah melewati proses normalisasi teks, kalimat tersebut berubah menjadi *output* "Yomi kalau tidak ada aku bersih bersih rumah terus" sesuai dengan entri pada kamus *slang* yang digunakan.

4.3.3. *Stopwords Removal*

```
from nltk.corpus import stopwords
import string, re, nltk

def remove_stopwords(text):
    stop_words = set(stopwords.words('indonesian'))
    words = nltk.word_tokenize(text)
    filtered_words = [word for word in words if word.lower() not in stop_words]
    return ' '.join(filtered_words)

text = "jangan kamu berharap deh"
print("Input: {}".format(text))
print("Output: {}".format(remove_stopwords(text)))
```

Gambar 4. 6 *Stopwords Removal* dalam Program

Berdasarkan gambar 4.6, pada tahapan ini dilakukan proses *filtering stopword* pada sebuah teks dengan menggunakan fungsi *remove_stopwords*. Fungsi ini menerima parameter teks, kemudian memecahnya menjadi kata per kata dengan menggunakan *nltk.word_tokenize()*. Setiap kata yang dihasilkan dicek apakah merupakan *stopword* dalam Bahasa Indonesia berdasarkan daftar *stopword* dari NLTK. Jika iya, kata tersebut dihapus, sementara kata-kata yang bukan *stopword* dimasukkan ke dalam daftar *filtered_words*. Daftar *filtered_words* yang sudah tidak mengandung *stopword*, kemudian digabungkan kembali dengan spasi di antara katanya menjadi sebuah kalimat tunggal yang dikembalikan oleh fungsi. Proses *filtering stopword* ini bertujuan untuk menghilangkan kata-kata umum yang tidak memiliki makna penting. Diharapkan dengan demikian, proses analisis selanjutnya dapat lebih terfokus pada kata-kata yang informatif setelah data terbebas dari *stopword*.

Tabel 4. 3 Contoh *Stopword Removal* Pada Dataset

Original Text	**Stopword Removal**
jangan kamu berharap deh	berharap deh

Tabel 4.3 tersebut menunjukkan proses *stopwords removal* dalam sebuah program.

Dalam contoh ini, *input* berupa kalimat "Yomi klo gak ada aku bebersih rumah mulu". Setelah melalui proses *stopwords removal*, *output* yang dihasilkan adalah "Yomi ada aku bersih bersih rumah terus". Kata "klo" dan "gak" dianggap sebagai *Stopwords* dan dihilangkan dari kalimat. Tujuan utama *stopwords removal* adalah untuk mengurangi dimensi atau kompleksitas data teks dengan membuang kata-kata yang tidak informatif. Hal ini dapat meningkatkan efisiensi dan akurasi dalam tugas-tugas seperti klasifikasi teks, pencarian informasi, atau pemodelan topik.

a. *Stemming*

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

def custom_stemming(text):
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()
    words = text.split()
    stemmed_words = [stemmer.stem(word) for word in words]

    stemmed_text = ' '.join(stemmed_words)
    return stemmed_text

text = "kedengarannya berlebihan sekali pernyataan bapak itu"
print("Input: {}".format(text))
print("Output: {}".format(custom_stemming(text)))
```

Gambar 4. 7 *Stemming* dalam Program

Berdasarkan gambar 4.7, pada tahapan ini dilakukan proses *stemming* dengan menggunakan aturan kustom pada sebuah kalimat menggunakan *library* Sastrawi. *Stemming* adalah proses mengubah kata berimbuhan menjadi kata dasarnya. Beberapa tahapan yang dilakukan yaitu pembuatan objek *stemmer* Sastrawi, definisi kamus kustom yang berisi pasangan kata asli dengan kata dasar yang diinginkan, pembuatan fungsi *custom_stemming* untuk melakukan *stemming* dengan aturan kustom yang didalamnya terdapat; kata diperiksa pada kamus kustom. Jika ada, maka kata tersebut diganti dengan kata dasar dari kamus dan jika kata tidak ada di kamus, dilakukan *stemming* bawaan Sastrawi. Dan tahapan yang terakhir yaitu penerapan fungsi *custom_stemming* pada

kalimat.

Tabel 4. 4 Contoh *Stemming* Pada Dataset

Original Text	**Stemming**
kedengarannya berlebihan sekali pernyataan bapak itu	dengar lebih sekali nyata bapak itu

Pada tabel 4.4 tersebut menampilkan *output* dari proses *stemming* dalam dataset. *Stemming* adalah proses menghilangkan imbuhan pada kata untuk mendapatkan kata dasarnya atau kata tunggal. Dalam contoh ini, kumpulan kata yang digunakan seperti "di tunggu" diubah menjadi bentuk kata dasarnya yaitu "tunggu", dan kata “kelanjuttannya” diubah menjadi bentuk baku yaitu “lanjut”. Dengan menghilangkan imbuhan, kata-kata dengan akar yang sama dapat dikelompokkan bersama, sehingga memudahkan dalam pencarian dan analisis teks.

b. *Pipeline*

Tabel 4. 5 *Output Pipeline* dalam Program

Input :	@Komentar_Tukiyem \gue suka banget naik gunung, cuman gue Gak PUNYA KAKI BANG457!! 😭 https://t.co/LqK2374f01
Output :	aku suka banget naik gunung cuman aku tidak punya kakak bang

Dari tabel 4.5 *output* program tersebut, menunjukkan *pipeline* program yang memproses data melalui beberapa tahap. Setiap tahap memiliki fungsi dan tujuan yang spesifik.

c. *Tokenizing FastText* & *Word2Vec*

```
import pandas as pd
from nltk.tokenize import word_tokenize
from tqdm.auto import tqdm

df = pd.read_csv('dataset_clean.csv')
df['token'] = df['clean'].apply(lambda x: word_tokenize(x.lower())
                                if isinstance(x, str) else [])
print(df[['token']])
```

Gambar 4. 8 *Tokenizing FastText* & *Word2Vec* dalam Program

Pada gambar 4.8, dilakukan tokenisasi pada kolom *clean* yang berisi teks ternormalisasi. Tokenisasi kata pada dataset teks menggunakan *library* Python NLTK (*Natural Language Toolkit*). Kode tersebut mengimpor modul tqdm dari paket *tqdm.auto* dan fungsi *word_tokenize* dari modul *nltk.tokenize*. Modul tqdm digunakan untuk menampilkan *progress bar* saat melakukan iterasi pada dataset, sementara fungsi *word_tokenize* bertanggung jawab untuk memecah teks menjadi token-token kata. Proses tokenisasi dilakukan dengan iterasi setiap baris pada kolom *clean* menggunakan *list comprehension*. Setiap kalimat dikonversi menjadi huruf kecil menggunakan metode *.lower()*, kemudian dilewatkan ke fungsi *word_tokenize* untuk memecahnya menjadi token-token kata. Hasil tokenisasi disimpan dalam daftar *sentences_fasttext*.

```
                                                 token
0    [sangat, di, tunggu, kelanjuttannya, tp, kalo,...
1                                            [astaga]
2    [rutinitas, anak, kos, yg, malas, gerak, ke, d...
3    [yomi, klo, tidak, ada, aku, bersih, rumah, mu...
4    [the, real, di, rumah, full, hari, benar, bena...
..                                                 ...
976  [kabar, dia, libat, dalam, buat, video, asusil...
977  [namun, dia, sendiri, ban, kabar, sebut, dan, ...
978  [tidak, cuma, itu, kawan, masih, banyak, lagi,...
979  [ada, gosip, tentang, si, hm, aktris, yang, ka...
980  [kabar, mereka, sering, temu, di, tempattempat...

[981 rows x 1 columns]
```

Gambar 4. 9 *Output Tokenizing FastText* & *Word2Vec* dalam Program

Dari gambar 4.9, *output* program tersebut merupakan proses tokenisasi atau pemisahan teks menjadi token-token individu. Hal ini terlihat dari daftar token yang ditampilkan, seperti ['sangat', 'di', 'tunggulah'], ['runtitas', 'anak', 'kos', 'ya', 'malas', 'gerak', 'ke', 'd...'], dan seterusnya. Setiap baris dalam *output* program merepresentasikan satu contoh atau data yang diproses. Misalnya, baris ke-0 menampilkan token ['sangat', 'di', 'tunggulah'], baris ke-2 menampilkan token ['runtitas', 'anak', 'kos', 'ya', 'malas', 'gerak', 'ke', 'd...'], dan seterusnya.

Hasil pada gambar diatas juga sama dengan pada tabel 4.6.

Tabel 4. 6 *Output Tokenizing FastText* & *Word2Vec* dalam Program

Hasil:

```
    token
0    [sangat, di, tunggu, kelanjuttannya, tp, kalo,...
1                                              [astaga]
2    [rutinitas, anak, kos, yg, malas, gerak, ke, d...
3    [yomi, klo, tidak, ada, aku, bersih, rumah, mu...
4    [the, real, di, rumah, full, hari, benar, bena...
..                                                 ...
976  [kabar, dia, libat, dalam, buat, video, asusil...
977  [namun, dia, sendiri, ban, kabar, sebut, dan, ...
978  [tidak, cuma, itu, kawan, masih, banyak, lagi,...
979  [ada, gosip, tentang, si, hm, aktris, yang, ka...
980  [kabar, mereka, sering, temu, di, tempattempat...
```

4.3.4. Proses Pemodelan (*Modelling*)

4.3.4.1. *FastText*

Berikut ini merupakan *pseudocode* dari pemodelan *Fasttext* yang dijabarkan pada tabel 4.10.

```
from gensim.models import FastText

model_fasttext = FastText(sentences_fasttext, vector_size=20,
                          window=5, min_count=1, sg=1,epochs=50)
model_fasttext.save('models/FASTTEXT/ft_models_FASTTEXT.wv')
```

Gambar 4. 10 *Train Model FastText* dalam Program

Pada gambar 4.10, kode mengimpor kelas *FastText* dari modul *gensim.models*. Selanjutnya, model *FastText* diinisialisasi dengan parameter-parameter tertentu, seperti *sentences_fasttext* (daftar kalimat yang akan dilatih), *vector_size*=50 (dimensi vektor kata), *window*=5 (ukuran jendela konteks), *min_count*=1 (kata dengan frekuensi minimal 1 akan dipertimbangkan), sg=1 (mode pelatihan *skip-gram*), dan *epochs*=50 (jumlah iterasi pelatihan). Objek model *FastText* disimpan dalam variabel *model_fasttext*. Setelah model diinisialisasi, kode menyimpan model yang telah dilatih ke dalam file dengan nama "models/*FASTTEXT*/ft_models_*FASTTEXT*.wv" menggunakan metode *.save()*. Pada

bagian selanjutnya, kode memuat kembali model yang telah disimpan sebelumnya ke dalam variabel *model_fasttext* menggunakan ekstensi .wv. Setelah model dimuat, kode menampilkan kunci-kunci (*keys*) yang terkandung dalam model menggunakan metode *.index_to_key*.

4.3.4.2. *Word2Vec*

Word2Vec juga lakukan dengan tahapan seperti yang dijabarkan pada *pseudocode* berikut ini.

```
from gensim.models import Word2Vec

model_word2vec = Word2Vec(sentences_word2vec,
                          vector_size=50, window=5,
                          min_count=1, sg=1,epochs=50)
model_word2vec.save('models/WORD2VEC/wv_models_WORD2VEC.wv')
```

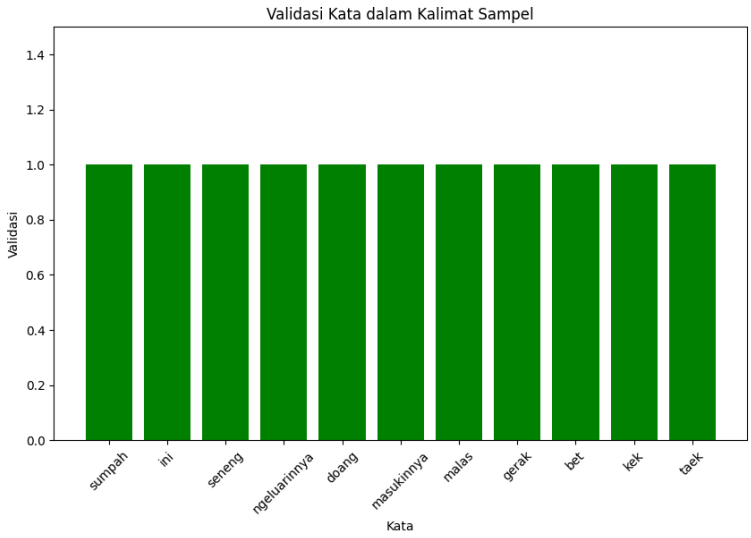
Gambar 4. 11*Train Model Word2Vec* dalam Program

Pada gambar 4.11, kode mengimpor kelas *Word2Vec* dari modul *gensim.models*. Selanjutnya, model *Word2Vec* diinisialisasi dengan parameter-parameter tertentu sama seperti *training* model *FastText* dalam program. Setelah model diinisialisasi, kode menyimpan model yang telah dilatih ke dalam file dengan nama "models/*WORD2VEC*/wv_models_*WORD2VEC*.vec" menggunakan metode *.save()*. Pada bagian selanjutnya, kode memuat kembali model yang telah disimpan sebelumnya ke dalam variabel *model_word2vec* menggunakan ekstensi .wv. Setelah model dimuat, kode menampilkan kunci-kunci (*keys*) yang terkandung dalam model menggunakan metode *.index_to_key*.

4.3.5. Validasi Vektor *Fastext*

Grafik di bawah menunjukkan hasil validasi kata dalam kalimat sampel "sumpah ini seneng ngeluarinnya doang masukinnya malas gerak bet kek taek" terhadap model FastText

yang telah dilatih. Semua kata dalam kalimat sampel (sumpah, ini, seneng, ngeluarinnya, doang, masukinnya, malas, gerak, bet, kek, taek) ditandai dengan warna hijau, menunjukkan bahwa semua kata ini ada dalam model FastText yang telah dilatih. Ini menunjukkan bahwa model telah berhasil mempelajari representasi vektor untuk semua kata dalam kalimat sampel dari dataset pelatihan. Sumbu y pada grafik menunjukkan validasi kata, di mana nilai 1 menunjukkan bahwa kata tersebut ada dalam model, dan nilai 0 menunjukkan bahwa kata tersebut tidak ada dalam model. Semua kata memiliki nilai validasi 1, menunjukkan bahwa mereka ada dalam model.

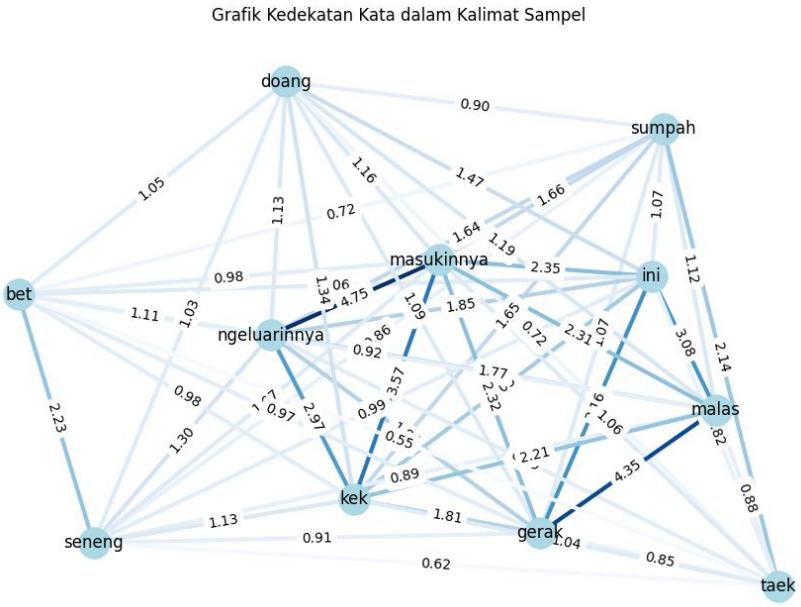


Gambar 4. 12 Pengujian Sampel Dari Dataset Untuk Validasi Vektor Model *FastText*

Kehadiran semua kata dalam model menunjukkan bahwa model FastText telah dilatih dengan dataset yang cukup besar atau beragam untuk menangkap variasi kata, termasuk kata slang atau kata-kata yang kurang umum. Model FastText memiliki kemampuan untuk menangani kata-kata yang tidak ada dalam model dengan lebih baik dibandingkan dengan Word2Vec karena FastText memecah kata menjadi n-gram. Dalam kasus ini, semua kata ada dalam model, menunjukkan bahwa pendekatan ini efektif. Validasi yang berhasil untuk semua kata menunjukkan bahwa model FastText yang dilatih memiliki cakupan yang baik

dan mampu mengenali berbagai kata dalam kalimat sampel.

Ini juga menunjukkan bahwa preprocessing data, seperti normalisasi dan tokenisasi, telah dilakukan dengan baik sehingga model dapat mengenali kata-kata dalam bentuk yang konsisten.



Gambar 4. 13 Pengujian Sampel Kalimat Melihat Semantik Setiap Kata Pada *FastText*

Grafik diatas menunjukkan grafik kedekatan kata dalam kalimat sampel “sumpah ini seneng ngeluarinnya doang masukinnya malas gerak bet kek taek” menggunakan model Word2Vec yang telah dilatih. Berikut adalah analisis dari grafik tersebut:

a. Setiap node dalam grafik mewakili kata dalam kalimat sampel. Setiap edge (garis penghubung) antara dua node menunjukkan kedekatan atau jarak antara dua kata berdasarkan vektor kata yang dihasilkan oleh model Word2Vec.

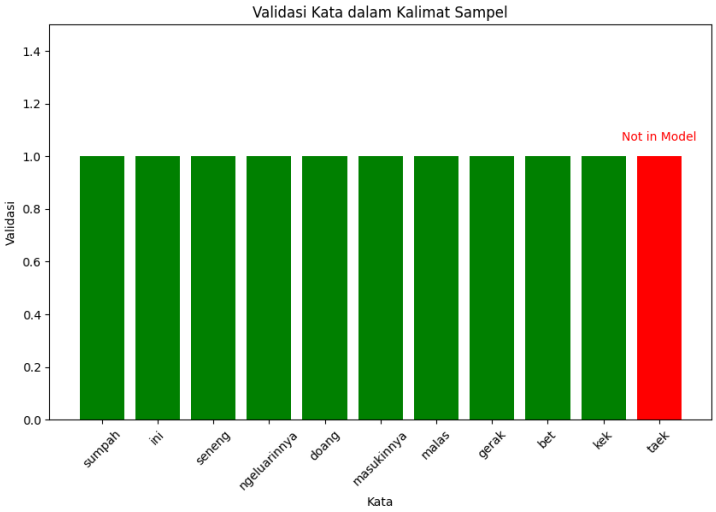
b. Bobot pada setiap edge menunjukkan jarak antara dua kata. Semakin kecil bobotnya, semakin dekat atau mirip dua kata tersebut dalam ruang vektor. Bobot yang lebih besar menunjukkan bahwa dua kata tersebut lebih jauh atau kurang mirip satu sama lain.

c. Kata "seneng" dan "kek" memiliki jarak yang sangat kecil (0.58), menunjukkan bahwa model menganggap kedua kata ini sangat mirip atau sering muncul dalam konteks yang serupa. Kata "seneng" dan "gerak" juga memiliki jarak yang kecil (0.59), menunjukkan kedekatan yang tinggi antara kedua kata ini. Kata "ngeluarinnya" dan "taek" memiliki jarak yang cukup besar (6.65), menunjukkan bahwa model menganggap kedua kata ini sangat berbeda atau jarang muncul dalam konteks yang serupa. Kata "doang" dan "taek" juga memiliki jarak yang besar (6.09), menunjukkan perbedaan yang signifikan antara kedua kata ini. Kata "taek" memiliki beberapa jarak yang besar dengan kata-kata lain, menunjukkan bahwa kata ini mungkin tidak sering muncul dalam dataset pelatihan atau memiliki konteks yang sangat berbeda dibandingkan dengan kata-kata lain dalam kalimat sampel. Kata "kek" memiliki beberapa jarak yang kecil dengan kata-kata lain seperti "seneng" (0.58) dan "gerak" (0.59), menunjukkan bahwa kata ini sering muncul dalam konteks yang mirip dengan kata-kata tersebut.

4.3.6. Validasi *Word2Vec*

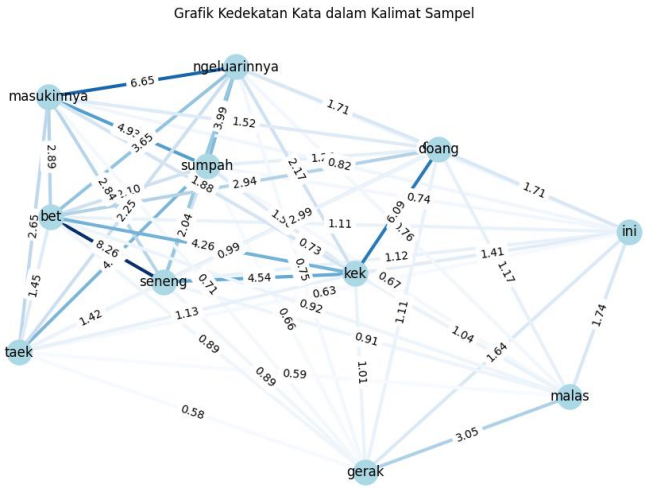
Grafik di atas menunjukkan hasil validasi kata dalam kalimat sampel "sumpah ini seneng ngeluarinnya doang masukinnya malas gerak bet kek taek" terhadap model Word2Vec yang telah dilatih. Kata "taek" ditandai dengan warna merah dan label "Not in Model", menunjukkan bahwa kata ini tidak ada dalam model Word2Vec yang telah dilatih. Ini bisa terjadi karena kata tersebut mungkin jarang muncul dalam dataset pelatihan atau merupakan kata slang yang tidak umum. Semua kata lainnya dalam kalimat sampel (sumpah, ini, seneng, ngeluarinnya, doang, masukinnya, malas, gerak, bet, kek) ditandai dengan warna hijau, menunjukkan bahwa kata-kata ini ada dalam model Word2Vec yang telah dilatih. Ini menunjukkan bahwa model telah berhasil mempelajari representasi vektor

untuk kata-kata ini dari dataset pelatihan. Sumbu y pada grafik menunjukkan validasi kata, di mana nilai 1 menunjukkan bahwa kata tersebut ada dalam model, dan nilai 0 menunjukkan bahwa kata tersebut tidak ada dalam model. Semua kata kecuali "taek" memiliki nilai validasi 1, menunjukkan bahwa mereka ada dalam model.



Gambar 4. 14 Pengujian Sampel Dari Dataset Untuk Validasi Vektor Model *Word2Vec*

Kehadiran kata "taek" yang tidak ada dalam model menunjukkan bahwa model mungkin perlu dilatih dengan dataset yang lebih besar atau lebih beragam untuk menangkap lebih banyak variasi kata, termasuk kata slang atau kata-kata yang kurang umum.



Gambar 4. 15 Pengujian Sampel Kalimat Melihat Semantik Setiap Kata Pada Word2Vec

Grafik diatas menunjukkan grafik kedekatan kata dalam kalimat sampel “sumpah ini seneng ngeluarinnya doang masukinnya malas gerak bet kek taek” menggunakan model Word2Vec yang telah dilatih. Berikut adalah analisis dari grafik tersebut:

a. Setiap node dalam grafik mewakili kata dalam kalimat sampel. Setiap edge (garis penghubung) antara dua node menunjukkan kedekatan atau jarak antara dua kata berdasarkan vektor kata yang dihasilkan oleh model Word2Vec.

b. Bobot pada setiap edge menunjukkan jarak antara dua kata. Semakin kecil bobotnya, semakin dekat atau mirip dua kata tersebut dalam ruang vektor. Bobot yang lebih besar menunjukkan bahwa dua kata tersebut lebih jauh atau kurang mirip satu sama lain.

c. Kata "seneng" dan "kek" memiliki jarak yang sangat kecil (0.58), menunjukkan bahwa model menganggap kedua kata ini sangat mirip atau sering muncul dalam konteks yang serupa. Kata "seneng" dan "gerak" juga memiliki jarak yang kecil (0.59), menunjukkan kedekatan yang tinggi antara kedua kata ini. Kata "ngeluarinnya" dan "taek" memiliki jarak yang cukup besar (6.65), menunjukkan bahwa model menganggap kedua kata ini sangat berbeda atau jarang muncul dalam konteks yang serupa. Kata "doang" dan "taek" juga memiliki jarak yang besar (6.09), menunjukkan perbedaan yang signifikan antara kedua kata ini. Kata "taek" memiliki beberapa jarak yang besar dengan kata-kata lain, menunjukkan bahwa kata ini mungkin tidak sering muncul dalam dataset pelatihan atau memiliki konteks yang sangat berbeda dibandingkan dengan kata-kata lain dalam kalimat sampel. Kata "kek" memiliki beberapa jarak yang kecil dengan kata-kata lain seperti "seneng" (0.58) dan "gerak" (0.59), menunjukkan bahwa kata ini sering muncul dalam konteks yang mirip dengan

kata-kata tersebut.

4.3.7. Hasil Evaluasi Perbandingan Model *FastText* dan *Word2Vec*

Setelah hasil dari pengujian didapat, pada tahapan dilakukan perbandingan dari skenario yang telah dilakukan. Hasil ringkasan dapat dilihat pada tabel 4.7.

Tabel 4. 7 Evaluasi Perbandingan Model *FastText* dan *Word2Vec*

Vektorisasi Dengan *FastText*	
Sampel Dokumen DF	**Kata-Kata tidak Tervektorisasi**
"aktif yuk tapi malas gerak"	-
Sampel Dokumen *Random*	**Kata-Kata tidak Tervektorisasi**
"Klo dari opini gw mah ya, yg bawa mobil itu gak salah tapi yg salah itu tukang parkirnya, malem2 pke baju hitam di tempat yg cahayanya kurang, awalnya juga gw kira itu cuma bayangan"	-
Vektorisasi Dengan *Word2Vec*	
Sampel Dokumen DF	**Kata-Kata tidak Tervektorisasi**
"aktif yuk tapi malas gerak"	-
Sampel Dokumen *Random*	**Kata-Kata tidak Tervektorisasi**
"Klo dari opini gw mah ya, yg bawa mobil itu gak salah tapi yg salah itu tukang parkirnya, malem2 pke baju hitam di tempat yg cahayanya kurang, awalnya juga gw kira itu cuma bayangan"	`Klo, opini, gw, ya,, mobil, gak, tukang, parkirnya,, malem2, pke, hitam, cahayanya, kurang,, awalnya, gw, bayangan`

Dari tabel 4.7, dilakukan evaluasi perbandingan antara model vektorisasi *FastText* dan *Word2Vec*. Evaluasi ini bertujuan untuk menganalisis kinerja kedua model dalam melakukan vektorisasi pada data teks. Proses evaluasi dilakukan dengan menggunakan dua jenis sampel dokumen, yaitu Dokumen DF (*Data Frame*) dan Dokumen *Random*. Pada vektorisasi dengan model *FastText*, baik Dokumen DF maupun Dokumen *Random* dapat divektorisasi dengan baik tanpa adanya kata-kata yang tidak tervektorisasi.

4.3.8. Hasil Evaluasi Kelebihan dan Kekurangan Model *FastText* dan *Word2Vec*

Tabel 4. 8 Kelebihan dan Kekurangan Model *FastText*

Kelebihan	**Kekurangan**

Mampu melakukan vektorisasi dengan baik pada semua kata dalam sampel dokumen DF (*Data Frame*) dan *Random* tanpa ada kata yang tidak tervektorisasi.	Mengambil kata – kata yang tidak bermakna
Penanganan kata *Out Of Vocabulary* (OOV)	Untuk waktu pemodelan didapatkan 0.432 detik lebih besar ketimbang *Word2Vec* yaitu 0.027 detik
Memiliki kemampuan untuk melakukan vektorisasi pada kata-kata yang tidak umum atau kata-kata baru.	Waktu normalisasi didapatkan 0.016 detik lebih besar ketimbang *Word2Vec* yaitu 0.006 detik
Kinerjanya konsisten dalam melakukan vektorisasi pada berbagai jenis data teks.	
Tidak mengalami kesulitan dalam melakukan vektorisasi pada kata-kata yang tidak terdapat dalam basis data pra-pelatihan.	

Tabel 4. 9 Kelebihan dan Kekurangan Model *Word2Vec*

Kelebihan	**Kekurangan**
Mampu melakukan vektorisasi dengan baik pada semua kata dalam sampel dokumen DF (*Data Frame*) tanpa ada kata yang tidak tervektorisasi.	Kesulitan menangani kata – kata tidak dikenal *Out Of Vocabulary* (OOV)
Word2Vec memiliki konsep yang sederhana dan mudah diimplementasikan, sehingga sering digunakan sebagai pilihan awal dalam tugas-tugas NLP.	Terdapat sejumlah kata yang tidak tervektorisasi, seperti "Klo", "opini", "gw", "ya", "mobil", "gak", "tukang", "parkirnya", "malem2", "pke", "hitam", "cahayanya", "kurang", "awalnya", dan "bayangan".
Model *Word2Vec* sedikit lebih cepat dalam proses normalisasi dibandingkan model *FastText*	Kinerjanya bergantung pada jenis data teks yang digunakan, yaitu kurang optimal pada data teks dengan kata-kata yang tidak umum atau tidak terdapat dalam basis data pra-pelatihan.
	Kurang konsisten dalam melakukan vektorisasi pada berbagai jenis data teks.

4.3.5. Pengujian *Cosine Similarity*

Telah dilakukan pencarian kata-kata yang memiliki kemiripan makna dengan kata 'astaga' menggunakan model *FastText* terlatih. Metode *similar_by_word ()* digunakan untuk mencari *top* 5 kata dengan tingkat kemiripan tertinggi berdasarkan *vector* representasi kata yang dihasilkan oleh *FastText*. Hasil pencarian kata serupa ditampilkan berupa pasangan

kata beserta skor kemiripannya. Semakin tinggi skor kemiripan menunjukkan vektor representasi kedua kata tersebut berada pada posisi yang berdekatan di ruang *vector*, yang merepresentasikan makna kata yang mirip berdasarkan konteks kalimat dari *corpus*. Proses pencarian kata serupa ini memanfaatkan kemampuan *FastText* dalam memetakan kata yang secara semantik memiliki makna yang dekat ke dalam *vector* representation yang berdekatan jaraknya. Dengan demikian dapat diketahui kata-kata yang berkaitan erat dengan suatu kata tertentu. Untuk lebih jelas dapat dilihat pada gambar 4.16.

```
similar_words = model_fasttext.similar_by_word('gua', topn=5)
# Print the similar words
print("Words similar :")
for word, similarity in similar_words:
    print(f"{word}: {similarity:.10f}")
```

✓ 0.0s

Gambar 4. 16 Pengujian *Cosine Similarity FastText* dalam Program

```
Words similar :
sisi: 0.8949731588
ngehindarin: 0.8910294771
suruh: 0.8902949691
males: 0.8771323562
wkwk: 0.8759982586
```

Gambar 4. 17 *Output* Pengujian *Cosine Similarity FastText* dalam Program

Dari gambar 4.17 *output* program, dapat diamati bahwa program tersebut melakukan pengujian terhadap kemiripan kata (*word similarity*) menggunakan metode *cosine similarity* pada model *FastText*. Secara spesifik, program menghitung nilai kemiripan *cosine* antara kata "naga" dengan empat kata lain, yaitu "gbsa", "piye", "hysterisih", dan "br77". Nilai kemiripan *cosine* berada dalam rentang 0 hingga 1, dengan nilai yang lebih tinggi menunjukkan kemiripan yang lebih besar antara dua kata. Berdasarkan *output* yang ditampilkan, kata "naga" memiliki nilai kemiripan *cosine* tertinggi dengan kata "gbsa", yaitu 0,9934859978. Sementara itu, nilai kemiripan *cosine* terendah adalah dengan kata

"hysterisih", yaitu 0,8928895496. Pengujian seperti ini penting dilakukan untuk mengevaluasi performa model *FastText* dalam menangkap kemiripan semantik antara kata-kata. Hasil pengujian dapat digunakan untuk memperbaiki model atau mengidentifikasi kekuatan dan kelemahan model dalam tugas-tugas tertentu.

```
similar_words = model_word2vec.similar_by_word('gua', topn=5)
# Print the similar words
print("Words similar :")
for word, similarity in similar_words:
    print(f"{word}: {similarity:.10f}")
```

Gambar 4. 18 Pengujian *Cosine Similarity Word2Vec* dalam Program

Dari gambar 4.18, telah dilakukan pencarian kata-kata yang memiliki kemiripan makna dengan kata 'gua' menggunakan model *Word2Vec* terlatih. Metode *most_similar ()* digunakan untuk mencari *top* 5 kata dengan tingkat kemiripan tertinggi berdasarkan *vector* representasi kata yang dihasilkan oleh *Word2Vec*. Hasil pencarian kata serupa ditampilkan berupa pasangan kata beserta skor kemiripannya. Semakin tinggi skor kemiripan menunjukkan vektor representasi kedua kata tersebut berada pada posisi yang berdekatan di ruang *vector*, yang merepresentasikan makna kata yang mirip berdasarkan konteks kalimat dari *corpus*. Proses pencarian kata serupa ini memanfaatkan kemampuan *Word2Vec* dalam memetakan kata yang secara semantik memiliki makna yang dekat ke dalam *vector representation* yang berdekatan jaraknya. Dengan demikian dapat diketahui kata-kata yang berkaitan erat dengan suatu kata tertentu.

```
Words similar :
sanggup: 0.9091110826
ngeditnya: 0.8381344676
kunci: 0.8260262609
ntr: 0.8131303787
bawa: 0.8122773170
```

Gambar 4. 19 *Output* Pengujian *Cosine Similarity Word2Vec* dalam Program

Dari gambar 4.19 *output* program, dapat diamati bahwa program tersebut menampilkan hasil perhitungan kemiripan kata (*word similarity*) antara beberapa kata atau istilah yang berbeda. Secara spesifik, program ini menghitung nilai kemiripan antara kata "bnget" dengan empat kata lainnya, yaitu "anak", "pulang", "pernah", dan "kak". Nilai kemiripan ini berada dalam rentang 0 hingga 1, dengan nilai yang lebih tinggi menunjukkan kemiripan yang lebih besar. Dari *output* yang ditampilkan, kata "bnget" memiliki nilai kemiripan tertinggi dengan kata "pulang", yaitu 0,9977979123. Sementara itu, nilai kemiripan terendah adalah dengan kata "kak", yaitu 0,0975351095.

4.3.6. Hasil Normalisasi Kata *Slang*

Setelah hasil dari pengujian dokumen data *frame* dan data *Random* didapat, pada tahapan ini dilakukan normalisasi kata *slang* menggunakan model *FastText* dan *Word2Vec*. Proses normalisasi kata *slang* dilakukan dengan melatih model *FastText* dan *Word2Vec* terlebih dahulu menggunakan data teks yang relevan. Setelah proses pelatihan selesai, kedua model tersebut kemudian digunakan untuk menormalisasi kata *slang* "mager" dalam serangkaian kalimat yang diberikan. Kinerja kedua model dalam melakukan normalisasi kata *slang* dievaluasi dengan membandingkan hasil normalisasi yang dihasilkan dengan bentuk kata yang diharapkan. Selain itu, waktu yang dibutuhkan untuk proses pelatihan model dan normalisasi kata juga dicatat untuk setiap model. Hasil penyajiannya dapat dilihat pada tabel 4.10.

Tabel 4.10. Perbandingan Normalisasi Kata *Slang* Menggunakan *FastText* dan *Word2Vec*

Model *FastText*		**Model *Word2Vec***	
Model time	***normalization Time***	***Model time***	***normalization Time***
0.432 *Secs*	0.016 *Secs*	0.427 *Secs*	0.006 *Secs*
Results		***Results***	
Origin	***Result***	***Origin***	***Result***
@sunmoonxzy sangat di tunggu kelanjuttannya, tp kalo sampe si kunti bogel msh berjaya auto mager bacanya ðŸ˜«	sangat di tunggu kelanjuttannya tp kalo sampai si kunti bogel msh jaya auto malas gerak baca	@sunmoonxzy sangat di tunggu kelanjuttannya, tp kalo sampe si kunti bogel msh berjaya auto mager bacanya ðŸ˜«	sangat di tunggu kelanjuttannya tp kalo sampai si kunti bogel msh jaya auto malas gerak baca
@tanyarlfes Rutinitas anak kos yg mager ke dapur	rutinitas anak kos yg malas gerak ke dapur	@tanyarlfes Rutinitas anak kos yg mager ke dapur	rutinitas anak kos yg malas gerak ke dapur
Yomi klo gak ada aku bebersih rumah mulu, klo ad aaku kok mager ya???? ðŸ˜€ðŸ˜€	yomi klo tidak ada aku bersih rumah mulu klo ad aaku kok malas gerak ya	Yomi klo gak ada aku bebersih rumah mulu, klo ad aaku kok mager ya???? ðŸ˜€ðŸ˜€	yomi klo tidak ada aku bersih rumah mulu klo ad aaku kok malas gerak ya
the real di rumah full seharian bener bener mager mau keluar ðŸ˜ https://t.co/38gGq5bRJV	the real di rumah full hari benar benar malas gerak mau keluar	the real di rumah full seharian bener bener mager mau keluar ðŸ˜ https://t.co/38gGq5bRJV	the real di rumah full hari benar benar malas gerak mau keluar
@thvvjjk_ doain gak mager bikinnya ðŸ«¡ðŸ¤£	doain tidak malas gerak bikin	@thvvjjk_ doain gak mager bikinnya ðŸ«¡ðŸ¤£	doain tidak malas gerak bikin

Berdasarkan tabel 4.10, hasil penelitian menunjukkan bahwa kedua model, *FastText* dan *Word2Vec*, mampu menormalisasi kata *slang* "mager" menjadi bentuk kata yang lebih baku, yaitu "malas gerak". Namun, terdapat perbedaan dalam waktu yang diperlukan oleh masing-masing model untuk melakukan proses normalisasi. Model *FastText* membutuhkan waktu 0,432 detik untuk proses pelatihan dan 0,016 detik untuk proses normalisasi kata *slang*. Sementara itu, model *Word2Vec* membutuhkan waktu yang lebih singkat, yaitu 0,027 detik untuk proses pelatihan dan 0,006 detik untuk proses normalisasi kata *slang*. Secara keseluruhan, kedua model menunjukkan kinerja yang baik dalam menormalisasi kata *slang* "mager". Namun, model *Word2Vec* sedikit lebih cepat dalam proses normalisasi dibandingkan dengan model *FastText*.

BAB V

PENUTUP

5.1. Kesimpulan

Penelitian ini telah berhasil mengeksplorasi dan membandingkan kinerja dari dua model *word embedding*, yaitu *FastText* dan *Word2Vec*, dalam melakukan normalisasi kata *slang* Bahasa Indonesia. Dari hasil evaluasi, *FastText* terbukti lebih unggul dalam menangani kata-kata tidak umum atau baru, sementara *Word2Vec* memiliki kesulitan dalam menangani kata-kata tidak dikenal. *FastText* juga lebih cepat dalam normalisasi, namun *Word2Vec* lebih konsisten dalam vektorisasi pada berbagai jenis data teks. Proses pencarian kata serupa menggunakan model *FastText* dapat membantu menemukan kata-kata yang memiliki makna mirip berdasarkan konteks kalimat dari corpus.

5.2. Saran

Untuk penelitian selanjutnya, disarankan untuk mengeksplorasi teknik-teknik lain dalam normalisasi kata *slang* Bahasa Indonesia, seperti pendekatan berbasis aturan (*rule-based*) atau kombinasi dengan metode pembelajaran mesin (*machine learning*). Selain itu, dataset yang lebih besar dan beragam dapat digunakan untuk meningkatkan kinerja model dan mencakup variasi kata *slang* yang lebih luas.

Selain itu, mempertimbangkan penggunaan model lain dan memperluas cakupan pengujian untuk memperoleh pemahaman yang lebih mendalam tentang penggunaan kata slang dalam bahasa Indonesia sehari-hari.

DAFTAR PUSTAKA

Adam, R. (2019, April 15). *Word Embedding* Bahasa Indonesia Menggunakan *Fasttext*.

Akbar, Y., & Sugiharto, T. 2023. Analisis Sentimen Pengguna *Twitter* di Indonesia Terhadap *ChatGPT* Menggunakan Algoritma *C4.5* dan *Naïve Bayes*. Jurnal Sains Dan Teknologi, 5 (1), 115–122. https://doi.org/10.55338/saintek.v4i3.1368

Budiman, I., Faisal, M. R., & Nugrahadi, T. D. 2020. Studi Ekstraksi Fitur Berbasis Vektor *Word2Vec* Pada Pembentukan Fitur Berdimensi Rendah. Jurnal Komputasi, Vol. 8 No. 1, 62–69.

Colab, G. (2022). *Google Colaboratory*.

Hannani, N. (2019, October 6). Pengertian *Twitter* Beserta Sejarah Dan Manfaat *Twitter* Yang Dibahas Secara Lengkap.

Istiqomah, D. S., Syifa Istiqomah, D., & Nugraha, V.(2018. Analisis Penggunaan Bahasa Prokem Dalam Media Sosial, 665 (5).

Joseph Lilleberg, Yun Zhu, & Yanqing Zhang. 2015. *Support Vector Machines And Word2vec For Text Classification With Semantic Features*. *In 2015 IEEE 14th International Conference On Cognitive Informatics Cognitive Computing* (ICCI*CC), 136–140.

Joulin Armand, Grave Edouard, Bojanowski Piotr, & Mikolov Tomas. 2016. *Bag Of Tricks For Efficient Text Classification.*

Kurniadi, D., Farisa, S., Haviana, C., & Novianto, A. 2020. Implementasi Algoritma *Cosine Similarity* Pada Sistem Arsip Dokumen Di Universitas Islam Sultan Agung. TRANSFORMTIKA, 17(2), 124–132.

Liddy, E. D. 2001. *Natural Language Processing*. *In Encyclopedia Of Library And Information Science* (2nd ed.). Marcel Decker, Inc.

Mihalcea, R., Corley, C., & Strapparava, C. 2006. *Corpus-Based And Knowledge-Based Measures Of Text Semantic Similarity*. *American Association for Artificial Intelligence*, Vol. 1, 775–780.

M.K.Vijaymeena, & K. Kavitha. 2016. *A Survey On Similarity Measures In Text Mining*. *Machine Learning and Applications: An International Journal* (MLAIJ), Vol. 3 (No. 1). https://doi.org/10.5121/mlaij.2016.3103.

Nurdin, A., Anggo, B., Aji, S., Bustamin, A., & Abidin, Z. 2020. Perbandingan Kinerja *Word*

Embedding Word2vec, *Glove*, Dan *Fasttext* Pada Klasifikasi Teks. Jurnal TEKNOKOMPAK, 14(2), 74.

Pradana, A. F. 2023. Perbandingan *Word Embedding Word2Vec*, *Glove*, Dan *Fasttext* Menggunakan *Deep Learning* Pada Ulasan Kondisi Pengguna Obat Kesehatan.

Ramadhanti, F., Wibisono, Y., & Ariani Sukamto, R. 2019. Analisis Morfologi Untuk Menangani *Out-Of-Vocabulary Words* Pada *Part-Of-Speech Tagger* Bahasa Indonesia Menggunakan *Hidden Markov Model*. *In* JLK (Vol. 2, *Issue* 1).

Riyaddulloh, R., & Romadhony, A. 2021. Normalisasi Teks Bahasa Indonesia Berbasis Kamus *Slang* Studi Kasus: *Tweet* Produk Gadget Pada *Twitter*. Agustus, 8 (4).

Rong, X. (2016). *Word2vec Parameter Learning Explained*.

Samudro, A. A. 2019. Normalisasi Teks Bahasa Indonesia Pada Media Sosial Berdasarkan *Fasttext Embeddings* Bahasa Indonesia *Text Normalization In Social Media Based On Fasttext Embeddings*.

Savrani, N. K. A., Eddyono, S., Kusumasari, B., Rajiyem, Widhyharto, D. S., Madya, S. H., Monggilo, Z. M. Z., Djindan, M., Putri, T. E., Larasati, Z. W., Prasetyo, W., Santoso, A. D., & Nityasari, A. 2021. *Big Data* Untuk Ilmu Sosial Antara Metode Riset Dan Realitas Sosial. *Gadjah Mada University Press* Anggota IKAPI dan APPTI.

Situmorang, L., Amalia, J., Hutahaean, E., & Angeli Sibarani, R. 2022. Membangun *Slang Dictionary* Untuk Normalisasi Teks Menggunakan *Pre-Trained Fasttext Model. In* Jurnal Jaringan Sistem Informasi Robotik (JSR) (Vol. 6, *Issue* 2). http://ojsamik.amikmitragama.ac.id.

Suminar, R. P. 2016. Pengaruh Bahasa Gaul Terhadap Penggunaan Bahasa Indonesia Mahasiswa Unswagati. Jurnal Logika , Vol XVII, No 3.

T I Sambi Ua, A. M., Lestriani, D. H., Sonia Kristanty Marpaung, E., Ong, J., Savinka, M., Nurhaliza, P., Yulia Ningsih, R., Kebun Jeruk Raya No, J., & Barat, J. 2023. Penggunaan Bahasa Pemrograman *Python* Dalam Analisis Faktor Penyebab Kanker Paru-Paru Universitas Bina Nusantara. Jurnal Publikasi Teknik Informatika (JUPTI), 2(2). https://doi.org/10.55606/jupti.v2i2.1742.

Utami, D. 2010. Karakteristik Penggunaan Bahasa Pada Status *Facebook*.

Vig, J., & Belinkov, Y. 2019. *Analyzing The Structure Of Attention In A Transformer Language Model*. http://arxiv.org/abs/1906.04284.

Yulianeu, A., & Oktamala, R. 2022. Sistem Informasi Geografis Trayek Angkutan Umum Di Kota Tasikmalaya Berbasis *Web*. Jurnal Teknik Informatika, Vol 10 No. 2, 125–134. https://doi.org/10.51530/jutekin.v10i2.669.

Yusuf, S., Fauzi, M. A., & Brata, K. C. 2018. Sistem Temu Kembali Informasi Pasal-Pasal KUHP (Kitab Undang-Undang Hukum Pidana) Berbasis *Android* Menggunakan Metode *Synonym Recognition* dan *Cosine Similarity* (Vol. 2, *Issue* 2). http://j-ptiik.ub.ac.id.

LAMPIRAN

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN SEMINAR HASIL SKRIPSI

engan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 19 APRIL 2024
Pukul	:	09:00 - 11:00
Tempat	:	RUANG PRODI

ah berlangsung Seminar Hasil Skripsi dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

nyatakan HARUS menyelesaikan perbaikan, yaitu:

- Pahami source code

8 - Mei - 2024

Dosen Pembimbing I,

MUHAMMAD FHADLI, S.Kom., M.Sc.
NIP. 199611232023211012

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN SEMINAR HASIL SKRIPSI

Dengan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 19 APRIL 2024
Pukul	:	09:00 - 11:00
Tempat	:	RUANG PRODI

telah berlangsung Seminar Hasil Skripsi dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

dinyatakan HARUS menyelesaikan perbaikan, yaitu:

perbaiki semua dari pengunji

Tobelo, 10/05/2024

Dosen Pembimbing II,

SYARIFUDDIN N. KAPITA, S.Pd., M.Si.
NIP. 199103122024211001

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN SEMINAR HASIL SKRIPSI

Dengan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 19 APRIL 2024
Pukul	:	09:00 - 11:00
Tempat	:	RUANG PRODI

telah berlangsung Seminar Hasil Skripsi dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

dinyatakan HARUS menyelesaikan perbaikan, yaitu:

1. Perbaiki semua kalimat yang typo, gambar yang tidak jelas dan referensi yang masih salah cek saat menghadap saya
2. Bab 1 perjelas lagi rumusan masalah dan batasan
3. Bab 3 metodologi dibaut sistematis
4. Bab 4 menjawab rumusan masalah
5. Bab 5 kesimpulan dan saran tolong di perbaiki.

Acc 15/5 2024

Dosen Penguji I,

Dr. ASSAF ARIEF, S.T., M.Eng.
NIP. 198307102008121001

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN SEMINAR HASIL SKRIPSI

Dengan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 19 APRIL 2024
Pukul	:	09:00 - 11:00
Tempat	:	RUANG PRODI

telah berlangsung Seminar Hasil Skripsi dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

8 5 2024

dinyatakan HARUS menyelesaikan perbaikan, yaitu:

- Download file naskah yg sdh saya beri catatan yang banyak
- Usul: Judul di ubah susunan katanya → list di naskah yg sy upload ke dan konsul ke pembimbing
- Rumusan masalah diperbaiki (lihat catatan di naskah)
- Tujuan penelitian diperbaiki
- Banyak penggunaan huruf kecil/kecil salah - Coba dicek perbaiki.
- Sistematika Penulisan
- Cantumkan Gambar dan keterangan gambarnya
- Tambahkan 1 sub bab di bab 4 yg membahas detail dr hasil evaluasi kelebihan dan kekurangan
- Kesimpulan belum menjawab tujuan dr apa yg dicari di penelitian ini.
- Cek Koreksi lain yg saya beri tanda pd naskah.

Dosen Penguji II,

Ir. AMAL KHAIRAN, S.T., M.Eng., IPM
NIP. 197401112003121003

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN SEMINAR HASIL SKRIPSI

Dengan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 19 APRIL 2024
Pukul	:	09:00 - 11:00
Tempat	:	RUANG PRODI

telah berlangsung Seminar Hasil Skripsi dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

dinyatakan HARUS menyelesaikan perbaikan, yaitu:

- aplikasi disesuaikan dengan rancangan pada BAB II
- Jadwal Penelitian di hapus
- Aplikasi di buat agr bisa digunakan oleh pengguna bisa berbasis web atau lainnya
- Hasil kedua algoritma harus lebih kecil dari penelitian sebelumnya
- Perbaiki penggunaan imbuat pada kaliamat, kata yang menujukan tempat tidak bisa di gabungkan "ditempat" disesuai dengan

ACC ok dan ujian tgl 6/05/2024

Dosen Penguji III,

YASIR MUIN, S.T., M.Kom.
NIDN. 9990582796

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN UJIAN SKRIPSI/TUTUP

Dengan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 31 MEI 2024
Pukul	:	09:00 - 10:30
Tempat	:	RUANG SIDANG

telah berlangsung Ujian Skripsi/Tutup dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

dinyatakan HARUS menyelesaikan perbaikan, yaitu:

- Pahami source code

28-6-2024
Acc

Dosen Pembimbing I,

MUHAMMAD FHADLI, S.Kom., M.Sc.
NIP. 199611232023211012

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN SEMINAR HASIL SKRIPSI

Dengan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 19 APRIL 2024
Pukul	:	09:00 - 11:00
Tempat	:	RUANG PRODI

telah berlangsung Seminar Hasil Skripsi dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

dinyatakan HARUS menyelesaikan perbaikan, yaitu:

perbaiki semua dari pengunji

Ternate, 01/07/2024

Dosen Pembimbing II,

SYARIFUDDIN N. KAPITA, S.Pd., M.Si.
NIP. 199103122024211001

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN UJIAN SKRIPSI/TUTUP

Dengan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 31 MEI 2024
Pukul	:	09:00 - 10:30
Tempat	:	RUANG SIDANG

telah berlangsung Ujian Skripsi/Tutup dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

dinyatakan HARUS menyelesaikan perbaikan, yaitu:

PERBAIKI SEMUA MASUKAN PENGUJI

ACC

27/10/2024

Dosen Penguji I

Dr. ASSAF ARIEF, S.T., M.Eng.
NIP. 198307102008121001

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN UJIAN SKRIPSI/TUTUP

Dengan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 31 MEI 2024
Pukul	:	09:00 - 10:30
Tempat	:	RUANG SIDANG

telah berlangsung Ujian Skripsi/Tutup dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

dinyatakan HARUS menyelesaikan perbaikan, yaitu:

- Tabel 4.6 dan Tabel 4.7 diperbaiki/digabung agar bisa memberikan dan menggambarkan perbandingan antara ke 2 metode.
- Bandingkan dg variabel yang sama

18/06 2024

Dosen Penguji II,

Ir. AMAL KHAIRAN, S.T., M.Eng., IPM
NIP. 197401112003121003

UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK
PROGRAM STUDI INFORMATIKA

DAFTAR PERBAIKAN UJIAN SKRIPSI/TUTUP

Dengan ini dinyatakan bahwa pada

Hari / tanggal	:	JUMAT, 31 MEI 2024
Pukul	:	09:00 - 10:30
Tempat	:	RUANG SIDANG

telah berlangsung Ujian Skripsi/Tutup dengan Peserta:

Nama Mahasiswa	:	RIFQAH NUR SURAYYA M. JEN
NPM	:	07352011097
Judul	:	NORMALISASI KATA SLANG BAHASA INDONESIA MENGGUNAKAN PERBANDINGAN MODEL FASTTEXT DAN WORD2VEC DENGAN PENDEKATAN NATURAL LANGUAGE PROCESSING

dinyatakan HARUS menyelesaikan perbaikan, yaitu:

- Penyajian Hasil di bab IV harus mudah dipahami, jangan masukan source code semua. bisa di ganti dengan grafik atau tabel yang bisa dimengerti
- Sitasi diperbaik ikut panduan style apa
- perbaiki daftar isi penomoran diperbaiki

ok.
tgl 20/6/24

Dosen Penguji III,

YASIR MUIN, S.T., M.Kom.
NIDN. 9990582796

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI
UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK PROGRAM STUDI INFORMATIKA
Kampus III Universitas Khairun, Kelurahan Jati Kota Ternate Selatan
http://if.unkhair.ac.id, http://unkhair.ac.id Group FB: if.unkhair

KARTU BIMBINGAN HASIL

Nama Mahasiswa : Rifqah Nur Surayya M.Jen
NIM : 07352011097
Dosen Pembimbing I : Muhammad Fhadli, S.Kom., M.Sc.
Judul : Perbandingan Normalisasi Kata *Slang* Bahasa Indonesia Menggunakan Model *Fasttext* & *Word2Vec* Dengan Pendekatan *Natural Language Processing*

NO	Tanggal	Uraian	Paraf
		Selesaikan Program	
		Perbaiki Program	
		Selesaikan bab 4 + 5	
		Pahami Program	
		Acc.	

KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI
UNIVERSITAS KHAIRUN
FAKULTAS TEKNIK PROGRAM STUDI INFORMATIKA
Kampus III Universitas Khairun, Kelurahan Jati Kota Ternate Selatan
http://if.unkhair.ac.id, http://unkhair.ac.id Group FB: if.unkhair

KARTU BIMBINGAN HASIL

Nama Mahasiswa : Rifqah Nur Surayya M.Jen
NIM : 07352011097
Dosen Pembimbing II : Syarifuddin N. Kapita, S.Pd., M.Si.
Judul : Perbandingan Normalisasi Kata *Slang* Bahasa Indonesia Menggunakan Model *Fasttext* & *Word2Vec* Dengan Pendekatan *Natural Language Processing*

NO	Tanggal	Uraian	Paraf
1)	12/02/2024	⊕ Samakan nama yg di proposal dengan program yg dibuat	
		⊕ Buat salah satu contoh hitungan manual eleni dan perbandingan yg digunakan	
		⊕ Buat hitungan manual di program	
2)	04/03/2024	⊕ Sesuaikan nama yg digunakan dengan hitungan manual di program	
3)	06/03/2024	⊕ Pg. Vektor Ambil 1 contoh untuk uji coba di Manual.	
		⊕ Masukkan di Bab 3	
		Ambon, 19/03/2024	