

**ANALISIS FEATURE EXTRACTION PADA TEXT
PROCESSING UNTUK ANALISIS SENTIMEN**

THESIS

**Oleh:
KURNIAWAN TRI PUTRA
NIM. 200605210019**



**PROGRAM STUDI MAGISTER INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023**

**ANALISIS FEATURE EXTRACTION PADA TEXT
PROCESSING UNTUK ANALISIS SENTIMEN**

THESIS

**Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Magister Komputer (M.Kom)**

**Oleh:
KURNIAWAN TRI PUTRA
NIM. 200605210019**

**PROGRAM STUDI MAGISTER INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGRI MAULANA MALIK IBRAHIM
MALANG
2023**

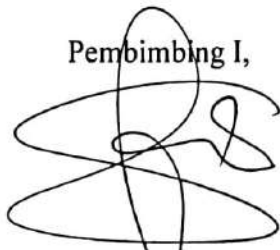
**ANALISIS FEATURE EXTRACTION PADA TEXT
PROCESSING UNTUK ANALISIS SENTIMEN**

THESIS

**Oleh:
KURNIAWAN TRI PUTRA
NIM. 200605210019**

Telah diperiksa dan disetujui untuk diuji:
Tanggal: 20 November 2023

Pembimbing I,



Dr. M. Amin Harivadi, M.T
NIP. 19670118 200501 1 001

Pembimbing II,



Dr. Cahyo Crvastian
NIP. 19740424 200901 1 008

Mengetahui,
Ketua Program Studi Magister Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang



Cahyo Crvastian
NIP. 19740424 200901 1 008

**ANALISIS FEATURE EXTRACTION PADA TEXT
PROCESSING UNTUK ANALISIS SENTIMEN**



THESIS

**Oleh:
KURNIAWAN TRI PUTRA
NIM. 200605210019**

Telah Dipertahankan di Depan Dewan Penguji Thesis
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Magister Komputer (M.Kom)
Tanggal: 27 November 2023

Susunan Dewan Penguji

Tanda Tangan

Penguji Utama	<u>Dr. Usman Pagalay, M.Si</u> NIP. 19650414 200312 1 001	()
Ketua Penguji	<u>Prof. Dr. Hj. Sri Harini, M.Si</u> NIP. 19731014 200112 2 002	()
Sekretaris Penguji	<u>Dr. M. Amin Hariyadi, M.T</u> NIP. 19670118 200501 1 001	()
Anggota Penguji	<u>Dr. Cahyo Crysdiان</u> NIP. 19740424 200901 1 008	()

Mengetahui dan Mengesahkan
Ketua Program Studi Magister Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang



Dr. Cahyo Crysdiان
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini:

Nama : Kurniawan Tri Putra
NIM : 200605210019
Program Studi : Magister Informatika
Fakultas : Sains dan Teknologi

Menyatakan dengan sebenarnya bahwa Thesis yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilalihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri,kecuali dengan mencantumkan sumber cuplikan pada daftar Pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan Thesis ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 20 November 2023

Yang membuat pernyataan,



Kurniawan Tri putra
NIM 200605210019

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Syukur alhamdulillah penulis hanturkan kehadiran Allah SWT yang telah melimpahkan Rahmat dan Hidayah-Nya, sehingga penulis dapat menyelesaikan studi di Program Studi Magister Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang sekaligus menyelesaikan Thesis ini dengan baik.

Selanjutnya penulis haturkan ucapan terima kasih seiring do'a dan harapan jazakumullah ahsanal jaza' kepada semua pihak yang telah membantu terselesaikannya Thesis ini. Ucapan terima kasih ini penulis sampaikan kepada:

1. Bapak Dr. M. Amin Hariyadi, M.T dan Bapak Dr. Cahyo Crysdiyan selaku dosen pembimbing Thesis, yang telah banyak memberikan pengarahan dan pengalaman yang berharga.
2. Segenap sivitas akademika Program Studi Magister Informatika, terutama seluruh Bapak/ Ibu dosen, terima kasih atas segenap ilmu dan bimbingan.
3. Ayahanda dan Ibunda tercinta yang senantiasa memberikan doa dan restunya kepada penulis dalam menuntut ilmu.
4. Semua pihak yang ikut membantu dalam menyelesaikan Thesis ini baik berupa materil maupun moril.

Penulis menyadari bahwa dalam penyusunan Thesis ini masih terdapat kekurangan dan penulis berharap semoga Thesis ini bisa memberikan manfaat kepada para pembaca khususnya bagi penulis secara pribadi.

Wassalamu'alaikum Wr. Wb.

Malang, 20 November 2023
Penulis,

DAFTAR ISI

HALAMAN SAMBUNG	i
HALAMAN PERSETUJUAN	ii
HALAMAN PENGESAHAN	iii
PERNYATAAN KEASLIAN TULISAN	iv
KATA PENGANTAR.....	v
DAFTAR ISI	vi
DAFTAR GAMBAR	viii
DAFTAR TABEL	x
ABSTRAK	xi
ABSTRACT	xii
مستخلص البحث	xiii
BAB I	1
PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Pernyataan Masalah.....	6
1.3 Tujuan Penelitian.....	6
1.4 Manfaat Penelitian	7
1.5 Ruang Lingkup Penelitian	8
BAB II.....	9
STUDI PUSTAKA.....	9
2.1 Analisis Sentimen.....	9
2.2 Feature Extraction	21
2.3 Kerangka Teori.....	26
BAB III.....	31
KONSEPTUAL MODEL DAN DATA PREPARATION	31
3.1 Konseptual Model	31
3.2 Data Preparation.....	34
3.2.1 Crawling.....	35
3.2.2 Labelling	37
3.2.3 Cleaning	38
3.2.4 Checking	44

BAB IV	48
ANALISIS SENTIMEN BERBASIS GLOVE DAN SVM	48
4.1 Deskripsi Penelitian.....	48
4.2 Import Data dan Library	49
4.3 Data Partition	50
4.4 Global Vectors for Word Representation (GloVe).....	51
4.5 Support Vector Machine (SVM)	57
4.6 Validation.....	69
BAB V.....	72
ANALISIS SENTIMEN BERBASIS FASTTEXT DAN SVM.....	72
5.1 Deskripsi Penelitian.....	72
5.2 Import data dan Library.....	73
5.3 FastText	74
5.3 Data Partition	79
5.5 Support Vector Machine (SVM)	80
5.6 Validation.....	92
BAB VI	95
PEMBAHASAN.....	95
6.1 Hasil Pengujian	95
6.2 Hasil Evaluasi	100
BAB VII	102
PENUTUP	102
7.1 Kesimpulan.....	102
7.2 Saran.....	103
DAFTAR PUSTAKA.....	104

DAFTAR GAMBAR

Gambar 2. 1	Alur analisis sentimen.....	9
Gambar 2. 2	Alur CRISP-DM.....	10
Gambar 2. 3	Alur analisis sentimen.....	12
Gambar 2. 4	Alur analisis sentimen.....	13
Gambar 2. 5	Alur Design Science Research (DSR)	15
Gambar 2. 6	Kerangka teori.....	26
Gambar 3. 1	Konseptual Model	31
Gambar 3. 2	Alur data preparation	34
Gambar 3. 3	Source code crawling data Twitter	36
Gambar 3. 4	Source code case folding	38
Gambar 3. 5	Source code remove punctuation	39
Gambar 3. 6	Source code stopwords removal	41
Gambar 3. 7	Source code stemming.....	42
Gambar 3. 8	Source code check and remove blank data	44
Gambar 3. 9	Hasil check and remove blank data.....	45
Gambar 3. 10	Source code check and remove duplicate data.....	45
Gambar 3. 11	Hasil check and remove duplicate data.....	46
Gambar 3. 12	Source code check imbalance data.....	47
Gambar 3. 13	Hasil check imbalance data.....	47
Gambar 4. 1	Analisis Sentimen GloVe dan SVM	48
Gambar 4. 2	Source code import library GloVe dan import data	49
Gambar 4. 3	Pembagian training data dan testing data	50
Gambar 4. 4	Source code split data training dan testing	50
Gambar 4. 5	Source code co-occurrence matrix	52
Gambar 4. 6	Source code proses vektorisasi kata	54
Gambar 4. 7	Hasil vektorisasi kata “cepat”	55
Gambar 4. 8	Source code representasi vektor berdasarkan setiap dokumen	55
Gambar 4. 9	source code menampilkan vektor GloVe	56
Gambar 4. 10	Hasil feature extraction GloVe.....	56
Gambar 4. 11	Algoritma Support Vector Machine (SVM).....	57
Gambar 4. 12	Source code tuning SVM.....	58

Gambar 4. 13	Hasil tuning SVM.....	61
Gambar 4. 14	Source code show best tuning.....	61
Gambar 4. 15	Output best tuning	61
Gambar 4. 16	Source code visualisasi params.....	65
Gambar 4. 17	Hasil visualisasi params.....	66
Gambar 4. 18	Source code visuliasasi hyperplane.....	67
Gambar 4. 19	Hasil hyperplane.....	68
Gambar 4. 20	Source code confusion matrix.....	69
Gambar 4. 21	Hasil tabel confusion matrix	70
Gambar 5. 1	Analisis sentimen FastText dan SVM.....	72
Gambar 5. 2	Source code import library FastText dan import data	73
Gambar 5. 3	Layer fasttext	74
Gambar 5. 4	Source code representasi bag of word.....	75
Gambar 5. 5	Hasil representasi bag of word.....	76
Gambar 5. 6	Source code rekontruksi data FastText.....	77
Gambar 5. 7	Source code penerapan softmax.....	78
Gambar 5. 8	Hasil akhir matrik.....	78
Gambar 5. 9	Pembagian training data dan testing data	79
Gambar 5. 10	Source code split data training dan testing	79
Gambar 5. 11	Algoritma Support Vector Machine (SVM).....	80
Gambar 5. 12	Source code tuning SVM.....	81
Gambar 5. 13	Hasil tuning SVM.....	84
Gambar 5. 14	Source code show best tuning.....	84
Gambar 5. 15	Output best tuning	84
Gambar 5. 16	Source code visualisasi params.....	88
Gambar 5. 17	Hasil visualisasi params.....	89
Gambar 5. 18	Source code visuliasasi hyperplane.....	90
Gambar 5. 19	Hasil hyperplane.....	91
Gambar 5. 20	Source code confusion matrix.....	92
Gambar 5. 21	Hasil tabel confusion matrix	93

DAFTAR TABEL

Tabel 2. 1 Hasil dari penelitian	17
Tabel 2. 2 Perbandingan teknik feature extraction.....	27
Tabel 2. 3 Perbandingan metode klasifikasi	29
Tabel 3. 1 Hasil crawling data Twitter.....	36
Tabel 3. 2 Hasil labeling menggunakan teknik crowdsourcing	37
Tabel 3. 3 Hasil case folding.....	39
Tabel 3. 4 Hasil remove punctuation.....	40
Tabel 3. 5 Hasil stopword removal.....	41
Tabel 3. 6 Hasil stemming	43
Tabel 4. 1 Hasil import data.....	49
Tabel 4. 2 Hasil co-occurrence matrix.....	53
Tabel 5. 1 Hasil import data.....	73
Tabel 6. 1 Perbandingan hasil confusion matrix	95
Tabel 6. 2 Perbandingan hasil performance matrix.....	97

ABSTRAK

Tri Putra, Kurniawan. 2023. Analisis Feature Extraction Pada Text Processing Untuk Analisis Sentimen. Thesis. Program Studi Magister Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. M. Amin Hariyadi, M.T (II) Dr. Cahyo Crysdiان.

Kata kunci : Analisis Sentimen, *Feature Extraction*, *Glove*, *Fasttext*, *Support Vector Machine*

Seiring pesatnya kemajuan perkembangan zaman mendorong peralihan digitalisasi, paling signifikan kita bisa rasakan sekarang ini adalah dalam bidang media sosial. Dengan banyaknya jumlah pengguna maka semakin banyak pula data yang tersimpan serta pemanfaatan yang paling tepat dan optimal adalah tuntutan yang harus terslesaikan. Dalam permasalahan tersebut langkah yang paling tepat adalah melakukan pemanfaatan data untuk tujuan analisis sentimen dengan membandingkan kinerja teknik *GloVe* dan *FastText* beracuan hasil *accuracy*, *precision*, *recall* dan *F1-score* serta pengaruh terhadap implementasi dari keduanya. Hasil menunjukkan bahwa teknik *FastText* mencapai kinerja lebih unggul dibandingkan dengan *GloVe*, dengan *accuracy* 85% dan *precision* 87%, sedangkan *GloVe* hanya mencapai 83% dan 85% secara berturut-turut. Meskipun *recall* model *FastText* hanya sedikit lebih tinggi, namun nilai *F1-score* yang mencapai 87% menunjukkan keseimbangan yang baik antara ketepatan dan keberhasilan dalam menemukan informasi. Selain itu, terdapat beberapa permasalahan dalam penerapan teknik *feature extraction* untuk analisis sentimen yaitu ukuran korpus teks yang besar, ketidakseimbangan kelas, dan sensitivitas *Support Vector Machine* terhadap skala fitur. Solusinya seperti subset data, *oversampling/undersampling*, dan normalisasi fitur dipergunakan untuk meningkatkan kinerja dan kehandalan model. Kesimpulan akhir dari penelitian ini mendukung penggunaan teknik *FastText* sebagai pendekatan yang lebih efektif dalam melakukan analisis sentimen dibandingkan dengan teknik *GloVe*.

ABSTRACT

Tri Putra, Kurniawan. 2023. Feature Extraction Analysis in Text Processing for Sentiment Analysis. Thesis. Master of Informatics Study Program, Faculty of Science and Technology, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Advisors: (I) Dr. M. Amin Hariyadi, M.T (II) Dr. Cahyo Crysdiان.

Keywords: Sentiment Analysis, *Feature Extraction*, *Glove*, *Fast text*, *Support Vector Machine*

Since the rapid development of the era encourages the transition of digitalization, the most significant change we can feel today is social media. With the large number of users, the more data are stored. Therefore, the most appropriate and optimal utilization become demands that must be resolved. To tackle this problem, the most appropriate step is using data for sentiment analysis purposes by comparing the performance of *GloVe* and *FastText* techniques based on the results of accuracy, precision, recall and *F1-score* as well as the effect on the implementation of both. The results showed that *the FastText* technique achieved superior performance compared to *GloVe*, with 85% accuracy and 87% precision. Meanwhile, *GloVe* only reached 83% and 85% respectively. Although the *FastText* model recall is only slightly higher, the *F1-score*, which is 87%, shows a good balance between accuracy and success in finding information. In addition, there are several problems in the application of feature extraction techniques for sentiment analysis that is the large size of the text corpus, class imbalances, and the sensitivity of the *Support Vector Machine* to feature scales. As the solutions, subsets of data, oversampling/undersampling, and normalization of features are used to improve the model performance and reliability. The final conclusion of this study supports the use of the *FastText* technique as a more effective approach in conducting sentiment analysis compared to the *GloVe* technique.

مستخلص البحث

تري بوترا، كورنياوان. ٢٠٢٣. تحليل استخراج الميزات في معالجة النصوص لتحليل المشاعر. البحث الجامعي. قسم المعلومات، كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: د. أمين هاردي، الماجستير. المشرف الثاني: د. جهيو كريسديان.
الكلمات الرئيسية: تحليل المشاعر، استخراج الميزة، *Fasttext*، *Glove*، آلة المنتجة الداعم.

نظرا لأن التقدم السريع في العصر يشجع على الانتقال إلى الرقمنة، فإن أهم ما يمكن أن نشعر به الآن هو في مجال وسائل التواصل الاجتماعي. مع العدد الكبير من المستخدمين، يتم تخزين المزيد من البيانات والاستخدام الأنسب والأمثل هو مطلب يجب حله. في هذه المشكلة، فإن الخطوة الأنسب هي استخدام البيانات لأغراض تحليل المشاعر من خلال مقارنة أداء تقنية *Glove* و *FastText* بناء على نتائج الدقة والثبات والاستدعاء ودرجة ف١ بالإضافة إلى التأثير على تنفيذ كليهما. أظهرت النتائج أن تقنية نص سريع حققت أداء فائقا مقارنة بفاز، بدقة ٨٥% وثبات ٨٧%، بينما حققت قفاز على ٨٣% و ٨٥% فقط على التوالي. على الرغم من أن استدعاء نموذج نص سريع أعلى قليلا فقط، إلا أن درجة ف١ البالغة ٨٧% تحقق توازنا جيدا بين الدقة والنجاح في العثور على المعلومات. بالإضافة إلى ذلك، هناك العديد من المشكلات في تطبيق تقنيات استخراج الميزات لتحليل المشاعر، وهي الحجم الكبير لمجموعة النص، وعدم التوازن الطبقي، وحساسية آلة المنتجة الداعم لمقياس الميزات. يتم استخدام حلول مثل بيانات المجموعة الفرعية، وأخذ العينات الزائدة / الناقصة، وتطبيع الميزات لتحسين أداء النموذج وموثوقيته. يدعم الاستنتاج النهائي لهذا البحث استخدام تقنية *FastText* كنهج أكثر فعالية في إجراء تحليل المشاعر مقارنة بتقنية *Glove*.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dewasa ini, seiring dengan pesatnya kemajuan perkembangan zaman serta pengaruh teknologi dan globalisasi merubah segala bentuk kebiasaan manusia secara perlahan-lahan beralih ke dunia digital secara menyeluruh. Pengaruh paling pesat dan kita bisa rasakan sekarang ini adalah dalam sektor media sosial, dengan fitur-fitur dan aplikasi yang ditawarkan media sosial seperti Instagram, Facebook, Twitter dll, memudahkan segala bentuk urusan dan pekerjaan manusia seperti dalam penyampaian berita, melakukan promosi bahkan sebagai sarana untuk menyalurkan emosi ataupun apa yang sedang dirasakannya. Dengan pengaruh yang ada, maka semakin banyak pengguna dan aktivitas dalam aplikasi media sosial tersebut semakin banyak juga data-data yang terekam dan tersimpan, kemudian terciptalah yang dikenal dengan istilah *big data*.

Menurut Dewi (2020) *big data* dapat diartikan sebagai kumpulan data yang berukuran sangat besar dan kompleks, data yang dihasilkan dari penggunaan media sosial tersebut juga termasuk kedalam kategori *big data*, yang mempunyai tiga karakteristik utama yaitu *volume* sebagaimana data-data tersebut jumlahnya sangat besar, *variety* sangat beragam dan *velocity* perubahan datanya sangatlah cepat. Adanya *big data* tersebut menjadi suatu hal yang sangat menguntungkan terutama bagi pelaku usaha, para pengembang bisnis, perusahaan besar serta pemerintahan yang memiliki cakupan luas. Data raksasa tersebut menjadi salah satu aset yang sangat berharga bagi mereka yang dapat mengelola dan memanfaatkan data dengan baik dan sebagaimana mestinya, hal ini sependapat dengan yang dikatakan presiden

Indonesia Ir H. Joko Widodo melalui pidato kenegaraan pada 16 Agustus 2019 di Istana Kepresidenan, beliau menuturkan bahwa “data adalah jenis kekayaan baru bagi negara kita, dan kini data lebih berharga daripada minyak”. Data berukuran besar ini bukan hanya berisi data-data penting saja namun juga bercampur dengan data tidak penting jika dilihat secara langsung, tetapi poin terpenting bukan pada apa isi data dan tingkat kepentingan data tersebut melainkan pada proses pengolahannya yang bisa menjadikan sesuatu yang sangat berharga. Teknik yang dapat digunakan untuk mengolah dan memanfaatkan *big data* tersebut adalah dengan teknik *text processing*.

Menurut Maryanto (2017) *text processing* merupakan teknik pengumpulan dan analisa data yang mempunyai tujuan untuk mendapatkan informasi penting didalamnya, teknik ini bekerja dengan menemukan hubungan yang berarti, pola dan kecenderungan dengan memeriksa dalam sekumpulan data besar, data yang tersimpan kedalam penyimpanan dengan menggunakan teknik pengenalan pola seperti teknik statistik dan matematik. *Text processing* dibagi menjadi beberapa kelompok lagi berdasarkan tugas yang dapat dilakukannya yaitu deskripsi, estimasi, prediksi, klasifikasi, pengklasteran dan asosiasi. Pada beberapa tugas yang dapat diselesaikan dengan teknik *text processing* tersebut, menurut hasil penelitian yang dilakukan oleh Permatasari et al. (2021) pada pemanfaatan teknik klasifikasi merupakan langkah yang paling tepat dihadapkan dengan data-data pada media sosial yang sebagian besar merupakan data berbentuk teks berupa cuitan atau postingan setiap user pengguna media sosial dengan melakukan analisis sentimen bertujuan mendapatkan pengetahuan maupun keputusan yang dapat dimanfaatkan oleh beberapa pihak terkait, agar data-data tersebut juga bermanfaat untuk semua.

Analisis sentimen pada pemanfaatan *big data* ini dengan cara kerja mencari dan mengolah data dari segala bentuk ekspresi atau keadaan yang sedang atau telah dialami seseorang *user* yang diluapkan dalam bentuk teks kedalam media sosial. Kemudian menurut Cindo et al. (2019) Analisis sentimen pada studi kasus data media sosial merupakan ilmu yang mencari emosi ataupun opini masyarakat berupa data berbentuk teks, dengan cara kerja mengelompokkan setiap pendapat masyarakat dan pengguna jasa terhadap perusahaan atau pemerintahan terkait pada media sosial, pendapat tersebut akan dilabeli atau dikelompokkan sesuai apa yang sedang dikeluhkan atau dibicarakan, label tersebut dapat berupa keputusan penentuan apakah komentar tersebut termasuk kedalam komentar saran ataupun hujatan yang sangat berguna sebagai bahan masukan serta evaluasi perusahaan maupun pemerintah untuk meningkatkan mutu pelayanannya.

Dalam pemrosesan data untuk analisis sentimen ini dengan cara kerjanya yaitu melakukan klasifikasi untuk menentukan sentimen seseorang dengan proses yang dikakukan oleh bantuan *machine* menjadi sangatlah sensitif dan harus sangat berhati-hati karena juga berkaitan dengan penentuan atau pelabelan diri seseorang yang dapat menimbulkan prasangka buruk. Larangan berprasangka buruk kepada sesama manusia sangat tidak diperbolehkan, yang juga dijelaskan dalam Al-Qur'an surat Al-Hujurat ayat 12 Allah berfirman:

يَا أَيُّهَا الَّذِينَ ءَامَنُوا اجْتَنِبُوا
كَثِيرًا مِّنَ الظَّنِّ إِنَّ بَعْضَ الظَّنِّ إِثْمٌ

Artinya: "Hai orang-orang yang beriman, jauhilah kebanyakan purba-sangka (kecurigaan), karena sebagian dari purba-sangka itu dosa."

Tafsir surat Al-Hujurat ayat 12: "Allah Subhanahu wa Ta'ala berfirman melarang hamba-hambanya dari banyak persangkaan, yaitu menuduh dan menganggap khianat kepada keluarga, kerabat dan orang lain tidak pada tempatnya. Karena sebagian dari persangkaan itu adalah dosa yang murni, maka jauhilah persangkaan tersebut dalam rangka kehati-hatian" (Ibnu Katsir, 7/291).

Dipahami dari nilai-nilai keislaman serta adab dalam bersosial diperlukan kehati-hatian dan ketelitian yang lebih untuk melakukan proses pemanfaatan pengolahan *big data* ini khususnya dalam hal analisis sentimen guna untuk menghindari perilaku atau kejadian prasangka buruk pada hasil penentuan label keputusan pada setiap komentar jika dilakukan dan di proses secara asal baik disengaja maupun tanpa sengaja. Langkah untuk melakukan pemrosesan secara tepat dan benar serta ketelitian yang dituntut dalam proses pengolahan data ini memerlukan sebuah mesin yang mempunyai kinerja tepat dan dapat melakukan prosesnya sendiri secara berkala tanpa melibatkan campur tangan manusia lagi, yaitu *machine learning*. Salahsatunya telah dilakukan penelitian sebelumnya oleh Hikmawan et al. (2020) dengan bahasan membandingkan beberapa metode-metode terkait klasifikasi teks pada analisis sentimen menggunakan *machine learning* guna untuk mendapatkan model terbaik dan hasil analisis sentimen yang akurat serta dapat dipercaya. Dengan artian pemilihan dan penerapan metode pada klasifikasi teks untuk analisis sentimen sangatlah penting dan perlu diperhatikan guna mewujudkan tujuan diawal yaitu dalam rangka kehati-hatian dan ketelitian dalam

melakukan pemrosesan guna menghindari buruk sangka serta mendapatkan *knowledge* atau pengetahuan seutuhnya dari *big data* yang telah didapatkan agar memang benar-benar berguna dan bisa dimanfaatkan.

Tidak cukup hanya dengan mencari serta mengimplementasikan sebuah metode yang paling optimal saja, tetapi permasalahan-permasalahan dalam proses ini masih ada yang harus diperhatikan dan sering terlewatkan. Bagian terpenting lain dan termasuk paling awal pada langkah pemrosesan yang juga harus diperhatikan adalah pada *feature extraction* walaupun terkadang pada penelitian-penelitian terkait pada tahap ini sering dianggap remeh dan disepelekan padahal dapat memberi dampak yang signifikan terhadap hasil modelling yang dibuat. Proses *feature extraction* ini berkerja sebelum data masuk kedalam modeling atau proses pengklasifikasian, langkah awal yang harus dilalui dengan data yang berupa teks atau bahasa manusia harus dikonversi menjadi numerik atau angka berbentuk vektor agar dapat diproses dan diolah komputer, langkah atau proses inilah yang dinamakan *feature extraction*. Seperti dituliskan Sabrila et al. (2021) tentang klasifikasi teks analisis sentimen menggunakan *machine learning* mengungkapkan bahwa pada proses *feature extraction* inilah yang sangat rumit dan sangat perlu diperhatikan dikarenakan banyaknya bahasa dan karakter gaya penulisan setiap orang yang berbeda beda dari berbagai suku dan negara yang ada, dalam melakukan proses pengestrakan kata mejadi bahasa yang dapat diterima dan di proses oleh komputer masih banyak kekurangan antara lain ketidakmampuannya memberikan informasi terkait makna, kesamaan kata, istilah bahasa dan kata yang belum pernah ditemui sebelumnya pada kamus. Daris inilah dapat dipahami bahwa proses *feature extraction* sangatlah penting dan harus diperhatikan terkait implementasi dan

pemilihannya yang akan diterapkan pada *machine learning* atau model yang akan dibuat guna mendapatkan hasil terbaik dan paling optimal dengan dihadapkan permasalahan-permasalahan yang ada pada proses klasifikasi teks.

1.2 Pernyataan Masalah

Pada proses pengolahan dan pemanfaatan *big data* dengan data berbentuk teks yang bersumber dari media sosial, dengan menerapkan teknik *text processing* menggunakan *machine learning* untuk analisis sentimen, masih banyak menemui permasalahan khususnya pada tahap *feature extraction*. Melihat pentingnya proses *feature extraction* serta permasalahan yang ada, penelitian ini membahas:

- a. *Feature extraction* apakah yang paling optimal dalam penerapan proses analisis sentimen dengan data berbentuk teks yang diukur menggunakan *accuracy*, *precision*, *recall* dan *f1-score*?
- b. Faktor apa saja yang dapat mempengaruhi kinerja dari penerapan *feature extraction* pada proses analisis sentimen?

1.3 Tujuan Penelitian

Berdasarkan latar belakang dan pernyataan masalah yang sudah dipaparkan, tujuan yang ingin di capai peneliti untuk menangani permasalahan yang ada dengan memaksimalkan proses analisis sentimen menggunakan *machine learning* khususnya pada tahap *feature extraction* ini adalah:

- a. Menganalisis proses penerapan teknik pada *feature extraction* yang tepat agar mendapatkan hasil yang paling optimal serta akurat dari berbagai perbandingan yang diterapkan untuk analisis sentimen.

- b. Mengidentifikasi permasalahan dan penanganan yang ada pada saat penerapan teknik *feature extraction* untuk analisis sentimen.

1.4 Manfaat Penelitian

Dengan target hasil dari penelitian ini untuk mendapatkan mesin pemrosesan analisis sentimen yang akurat dan dapat dipercaya juga dapat bermanfaat bagi beberapa pihak terkait yaitu:

- a. Perusahaan penyedia layanan ekspedisi JNT Express
 - Hasil dari analisis ini dapat digunakan perusahaan untuk melakukan pekerjaan secara tepat dan cepat juga proses dapat dilakukan secara mandiri, dihadapkan dengan jangkauan segmen perusahaan yang sangat besar dan luas untuk memangkas biaya operasional dan waktu.
 - Cuitan hasil analisis dapat memberikan wawasan tentang aspek-aspek tertentu dari layanan atau kinerja yang perlu ditingkatkan, perusahaan dapat menggunakan informasi ini untuk merancang strategi perbaikan operasional dan meningkatkan kualitas kepuasan pelanggan kedepannya.
- b. Masyarakat pengguna layanan ekspedisi JNT Express
 - Dengan memberikan umpan balik melalui media sosial, masyarakat dapat mempengaruhi perusahaan untuk meningkatkan kualitas layanan dan respons terhadap kebutuhan konsumen. Hal ini berpotensi memberikan pengalaman pengguna yang lebih baik bagi semua pelanggan.
 - Masyarakat dapat merasa lebih diberdayakan ketika perusahaan merespon dan bertindak atas umpan balik mereka melalui media sosial. Penelitian

analisis sentimen pada data berbentuk teks dari cuitan Twitter ini memberikan konsumen platform untuk menyampaikan suara mereka dengan tepat dan cepat, yang dapat mendorong perusahaan untuk bertanggung jawab dan lebih berkomitmen terhadap pelayanan kedepannya.

1.5 Ruang Lingkup Penelitian

Dalam ruang lingkup penelitian ini, agar tidak menyimpang dari pokok bahasan dan yang dibahas lebih spesifik serta berfokus kepada tema penelitian akan diberikan batasan yang jelas yaitu:

- a. Sumber data yang akan diambil adalah berasal dari aplikasi media sosial bernama Twitter.
- b. Data yang diambil adalah data berupa teks cuitan ataupun postingan yang dikelompokkan berdasarkan *hashtag* (#) dengan topik perusahaan atau pemerintahan yang akan di analisis.
- c. Waktu priode pengambilan data berdasarkan postingan yang akan diambil adalah dari 11 April 2020 sampai dengan 23 April 2020 dengan jumlah data sebanyak 1000 data.
- d. Objek dari penelitian ini adalah perusahaan bergerak dibidang ekspedisi dan juga menjadi salah satu perusahaan penyandang status *unicorn* di Indonesia dikarenakan segmennya yang sangat luas yaitu J&T Express.

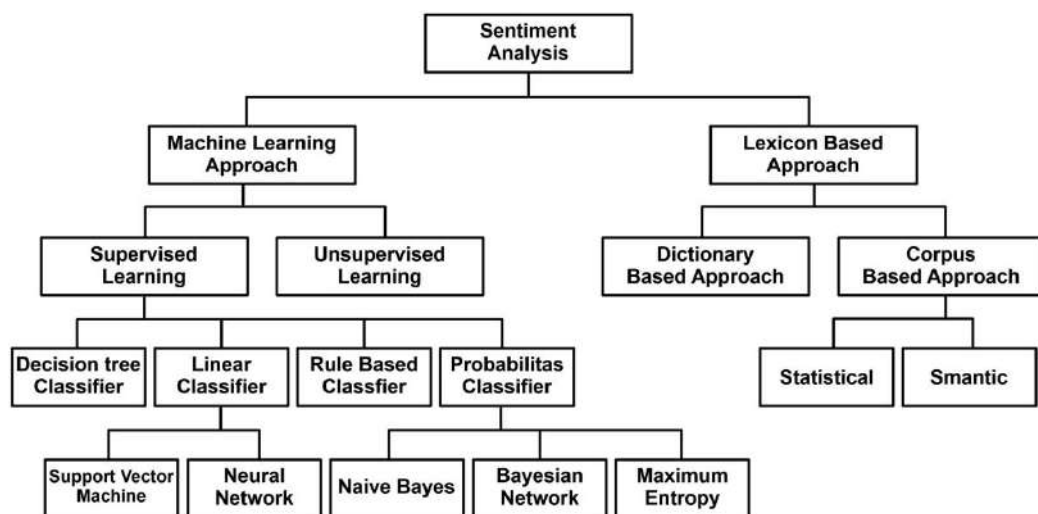
BAB II

STUDI PUSTAKA

Bagian bab ini membahas tentang penelitian terdahulu yang terkait atau memiliki tema penelitian yang sama dengan penelitian yang akan dilakukan yaitu penanganan permasalahan pada proses klasifikasi teks analisis sentimen menggunakan *machine learning*, dari berbagai sumber yang telah diuji kebenarannya seperti jurnal penelitian, laporan penelitian dan terbitan buku.

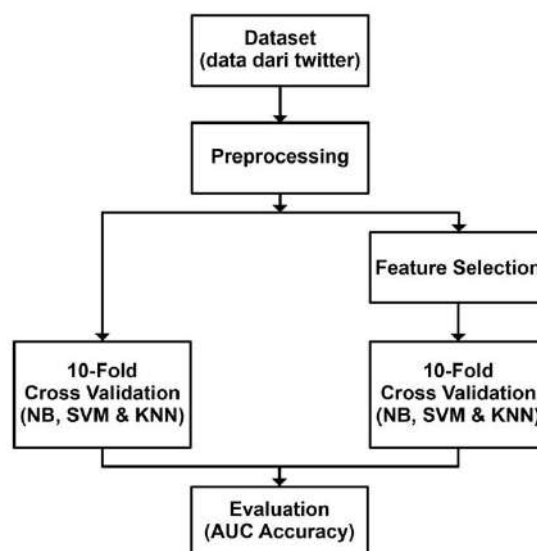
2.1 Analisis Sentimen

Pada paparan dari studi pustaka kali ini mengulas alur proses penelitian yang diterapkan pada analisis sentimen. Peneliti mengutip dalam penelitian yang ditulis oleh Jagdale et al. (2019) pada penelitian ini menjelaskan bahwa proses analisis sentimen mengacu pada studi analisis teks dan pemrosesan bahasa alami agar dapat diterjemahkan dan juga dapat dimengerti oleh *machine* dengan cara kerja mengidentifikasi, mengekstrak dan mempelajari informasi subjektif dari data tekstual secara ilmiah, kemudian tugas terpenting dari analisis sentimen ini adalah untuk menentukan polaritas teks sikap pelanggan, dengan alur berikut.



Gambar 2. 1 Alur analisis sentimen

Pada penelitian selanjutnya terkait alur dan langkah-langkah proses analisis sentimen yang dilakukan oleh Giovani et al. (2020) pada penelitiannya ini melakukan analisis sentimen kepada aplikasi ruang guru, guna mengetahui keberhasilan suatu aplikasi atau pemrosesan yang sedang ditelitinya. Dalam proses klasifikasi analisis sentimen ini dengan mengambil data dari aplikasi media sosial Twitter dengan menerapkan beberapa metode dan teknik validasi yang dilakukan. Berdasarkan sumber penelitian tersebut peneliti hanya berfokus untuk mengambil bagian dari alur atau langkah-langkah dari penelitian yang telah dilakukan untuk analisis sentimen dari awal dilakukannya penelitian hingga memperoleh hasil dari penelitian tersebut. Alur penelitian yang dilakukannya ini mengadopsi dari model *Cross-Industry Standard Process for Data Mining (CRISP-DM)*, berikut merupakan alur atau langkah bagan dari penelitian yang telah dilakukan juga disertai dengan paparannya secara rinci:

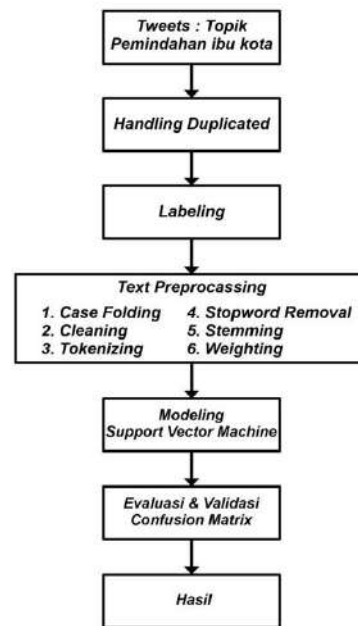


Gambar 2. 2 Alur CRISP-DM

Gambar 2.2 merupakan langkah-langkah proses klasifikasi analisis sentimen, pada penelitian ini juga akan menjelaskan poin setiap langkahnya sebagai berikut:

1. *Business understanding* : meneliti analisis sentimen aplikasi ruangguru dengan alur proses *10-fold cross validation*;
2. *Data understanding* : *crawling* data menggunakan *RapidMiner* dengan *query* “ruangguru” data yang sudah didapat berformat *.xlsx*;
3. *Data preparation (Preprocessing)* : menyiapkan data dan membersihkan data dengan teknik *transform case, remove http, tokenize, filter tokens by length* dan *remove stopwords*;
4. *Modeling* : pada tahap ini dengan menerapkan metode yang sudah dipersiapkan kedalam proses klasifikasi analisis sentimen seperti *Support Vector Machine, Naïve Bayes, K-Nearest Neighbor* dll.
5. *Evaluation* : melakukan uji validasi atau uji keberhasilan pada proses analisis sentimen yang sudah dilakukan dengan teknik *10-fold cross validation*.

Kemudian pada penelitian selanjutnya yang telah dilakukan oleh Arsi & Waluyo (2021) menurutnya klasifikasi teks analisis sentimen menggunakan *machine learning* adalah proses mengekstraksi, memahami dan mengolah data berupa teks yang tidak terstruktur secara otomatis kedalam bentuk angka atau nilai yang dapat diproses oleh komputer guna mendapatkan informasi penting berupa sentimen yang terdapat pada sebuah teks berbentuk kalimat tersebut yang berupa pendapat atau opini untuk dapat dimanfaatkan oleh beberapa pihak terkait. Dalam penerapan proses analisis sentimen menggunakan metode klasifikasi yaitu SVM (*Support Vector Machine*) dengan penelitian yang dilakukan menerapkan beberapa alur atau langkah-langkah pemrosesan seperti yang terbagi atas beberapa level atau fase yang akan dimulai dari mempersiapkan data yang akan digunakan untuk penelitian hingga mendapatkan hasil yang diinginkan, sebagai berikut:

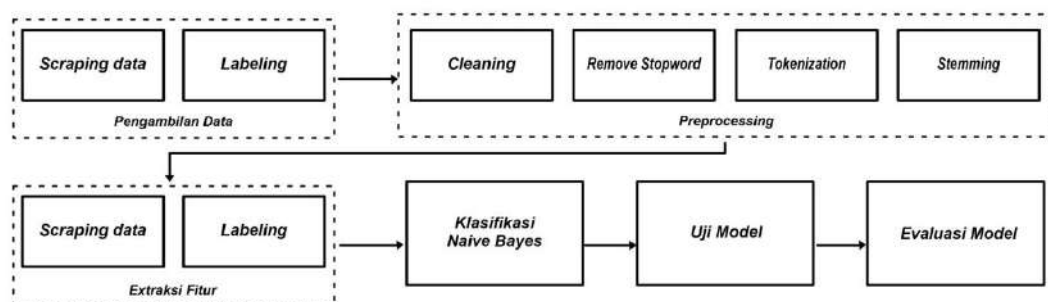


Gambar 2. 3 Alur analisis sentimen

Gambar 2.3 merupakan langkah-langkah proses analisis sentimen, pada penelitian ini juga menjelaskan poin setiap langkahnya sebagai berikut:

1. *Dataset* : adalah data mentah, digunakan data *tweet* terkait topik pemindahan ibu kota indonesia yang diambil dengan metode pada API Twitter;
2. *Crawling* : merupakan teknik mengumpulkan data pada sebuah *website*;
3. *Duplicate* : pembuangan data bertumpuk memiliki kesamaan lebih dari satu;
4. *Labeling* : pemberian label pada setiap data secara manual oleh orang yang berkompeten dibidangnya;
5. *Preprocessing* : pengolahan data mentah menjadi koleksi data yang siap digunakan dengan cara membersihkan teks dari karakter tidak penting;
6. *Training dan Testing* : pembagian data yang akan dilatih dan data yang diuji;
7. *Modeling* : diproses inilah peran metode yang dipilih melakukan pemrosesan;
8. *Validasi dan Evaluasi* : Mengukur dan mengetahui sejauhmana keberhasilan model yang telah dilakukan:

Selanjutnya penelitian yang dilakukan oleh Imron (2019) dalam laporan skripsinya, tujuan dilakukannya ini adalah dengan menemukan sentimen untuk melihat pandangan atau pendapat teks yang berkaitan terhadap sebuah masalah atau objek, apakah cenderung berpandangan positif atau negatif. Dalam penerapan proses analisis sentimen menggunakan algoritma klasifikasi yaitu *Naïve Bayes Classifier* yang diterapkan dengan *machine learning* terdapat beberapa alur atau langkah-langkah pemrosesan seperti berikut:



Gambar 2. 4 Alur analisis sentimen

Gambar 2.4 merupakan langkah-langkah proses analisis sentimen mulai dari pengambilan data awal hingga proses evaluasi hasil, pada penelitian ini juga menjelaskan poin pada setiap langkah-langkahnya sebagai berikut:

1. *Pengambilan Data* : mendapatkan data komentar dari situs *Tripadvisor* dan aplikasi Facebook dengan menggunakan teknik *crawling* dan *labeling*;
2. *Preprocessing* : menyeleksi data dan mengubahnya menjadi data yang terstruktur. Pada tahap *preprocessing* terdiri dari 4 tahapan, yaitu *cleaning*, *remove stopword*, *tokenization* dan *stemming*;
3. *Ekstraksi Fitur* : pembuatan fitur untuk mempermudah jalannya proses *learning* atau sering disebut proses *feature extraction* dengan cara merubah bahasa alami manusia agar dapat diproses dan dimengerti oleh komputer;

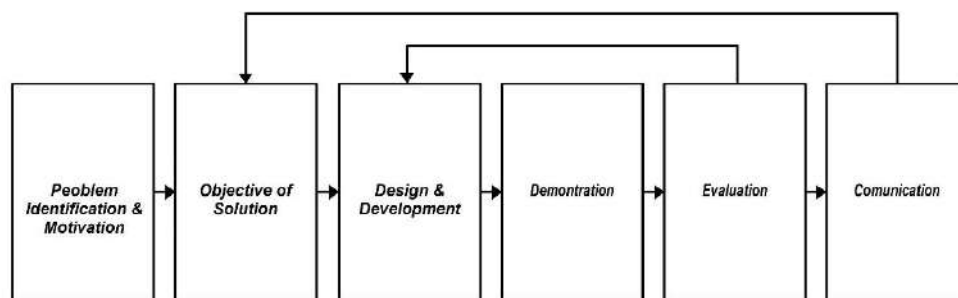
4. *Klasifikasi Naïve Bayes* : pada tahap ini dimulainya proses pengklasifikasian berdasarkan sentimen yang ada di dalam dokumen;
5. *Uji Model* : untuk mengukur nilai performa modeling yang telah dilakukan.;
6. *Evaluasi Model* : dengan cara melihat tingkat akurasi metode melalui *confusion matrix* dan tabel *accuracy* serta *precision* untuk tiap model:

Pada penelitian ini yang dilakukan oleh Sari & Wibowo (2019) menurutnya *sentiment analysis* atau analisis sentimen adalah suatu teknik mengekstrak data teks untuk mendapatkan informasi tentang sentimen bernilai positif maupun negatif. Dalam peenelitiannya ini melakukan analisis sentimen terhadap JD.id dengan menggunakan teknik *text mining* metode klasifikasi, akan diketahui suatu sentimen bernilai positif atau negatif. Kemudian langkah-langkah dalam penelitian yang telah dibuat untuk dapat memenuhi prosedur agar hasil yang didapatkan dapat sesuai dengan tujuan sebagai berikut:

1. Identifikasi masalah, tujuan penelitian dan studi literatur
2. Pengumpulan data *tweet* pada Twitter
3. Pengolahan data menggunakan *text preprocessing*
4. Pembobotan TF-IDF
5. Klasifikasi menggunakan *Naïve Bayes* dan penambahan fitur konversi *icon*
6. Validasi dan evaluasi

Seperti langkah-langkah yang sudah disebutkan, mulai dari identifikasi masalah, pengumpulan data, pengolahan data, pengekstrakan teks menjadi angka berupa vektor, penerapan metode klasifikasi teks sampai dengan validasi dan evaluasi merupakan runtutan proses dari penelitian *text mining* menggunakan metode klasifikasi yang sudah dirancang secara runtut dan sesuai prosedur yang berlaku:

Pada penelitian selanjutnya oleh Putri et al. (2022) dengan melakukan analisis sentimen masyarakat terhadap kinerja Dewan Perwakilan Rakyat (DPR) yang diungkapkan melalui media sosial Twitter. Ada beberapa tahap untuk melakukan analisis sentiment menurutnya, yaitu yang pertama adalah pengumpulan data *crawling*, kemudian *preprocessing* data yang terdiri dari proses *cleaning data*, *tokenization*, *stopword removal* dan *case folding*, *splitting* data dan klasifikasi teks atau penerapan metode-metode yang dapat diterapkan pada klasifikasi pada studi kasus ini adalah menggunakan metode klasifikasi *Naive Bayes Classifier* dengan menggunakan model penelitian *Design Science Research (DSR)* yang telah diterapkan dengan alur sebagai berikut:



Gambar 2. 5 Alur Design Science Research (DSR)

Dari alur *Design Science Research (DSR)* yang sudah digambarkan merupakan langkah-langkah proses runtutan pembelajaran mulai Identifikasi masalah yang ada di lapangan, menentukan objek solusi dari permasalahan yang telah dirumuskan, melakukan perancangan dan pengembangan sistem guna untuk memecahkan permasalahan, *demonstrasi system* atau pengujian system dengan tujuan untuk mengetahui apakah system yang telah dibuat sudah berjalan sesuai dengan rancangan sebelumnya dan terakhir adalah tahap evaluasi yaitu untuk memberikan ulasan dari hasil pengujian yang telah dilakukan pada sistem.

Selanjutnya sebagai dasar dan pembukaan dari studi pustaka analisis sentimen kali ini, peneliti mengutip dalam buku yang ditulis oleh Wahyono (2018), sebelum mengenal apa itu analisis sentimen lebih dalam lagi kita disuguhkan dengan kata “*machine learning*” yang merupakan salah satu ilmu dari cabang kecerdasan buatan, khususnya yang mempelajari tentang bagaimana komputer mampu belajar dari data untuk meningkatkan kecerdasannya. Salah satu teknik yang dilakukan dalam proses *machine learning* adalah *supervised learning* dimana teknik ini menggunakan data latih untuk melakukan pembelajaran pada mesin data yang dimaksud adalah data yang sudah berlabel. Tujuan dari teknik *supervised learning* ini adalah agar mesin mampu mengidentifikasi label *input* yang baru dengan menggunakan fitur yang dimilikinya untuk melakukan klasifikasi, sedangkan klasifikasi adalah penyusunan suatu data secara sistematis dengan aturan dan kaidah yang sudah diterapkan yang membutuhkan metode untuk memprosesnya. Beberapa metode yang biasa sering digunakan dalam proses klasifikasi tersebut diantaranya adalah *K-Nearest Neighbor*, *Decision tree*, *Naïve Bayes* dan *Support Vector Machine*.

Kemudian pada penelitian yang telah dilakukan oleh Wibawa et al. (2018) dalam penelitiannya yang membahas berbagai macam perbandingan metode-metode klasifikasi teks pada analisis sentimen yang umum digunakan yaitu *NB*, *SVM*, *DT* dan *KNN* serta menjabarkan kelebihan dan kekurangan dari masing-masing metode yang dihasilkan dari beberapa ilmu yang telah diterapkannya, pada penerapan penelitian yang telah dilakukan dengan hasil dan kesimpulan yang didapatkan serta dapat ditarik evaluasi untuk kelebihan dan kekurangan dari masing-masing metode yang telah diterapkan dan diteliti, sebagai berikut:

Tabel 2. 1 Hasil dari penelitian

Algoritma	Kelebihan	Kekurangan
<i>Naïve Bayes</i>	<ul style="list-style-type: none"> - Kinerja masih tetap unggul ketika pengujian dilakukan pada tipe data kategori. - Semua atribut tidak saling ketergantungan yang diberikan oleh nilai pada variabel kelas. - Performa Baik. 	<ul style="list-style-type: none"> - Sensitif pada fitur yang terlalu banyak, sehingga membuat akurasi menjadi rendah. - Ukuran dari vektor fitur yang dihasilkan cukup besar dan butuh teknik untuk memperkecil ukuran vektor.
<i>Support Vector Machine</i>	<ul style="list-style-type: none"> - Mengklasifikasikan suatu <i>pattern</i>, yang tidak termasuk data yang dipakai dalam fase pembelajaran metode itu. - Mempunyai banyak pilihan teknik lagi didalamnya. - Dimplementasikan dengan mudah. 	<ul style="list-style-type: none"> - Sulit dipakai dalam problem berskala besar. Skala besar dalam hal ini dimaksudkan dengan jumlah sampel yang diolah. - Jenis kernel <i>Support Vector Machine</i> berpengaruh pada akurasi sistem.
<i>Decission Tree</i>	<ul style="list-style-type: none"> - Data lebih akurat. - Meningkatkan efisiensi komputasi. - Menghindari hilangnya atribut. 	<ul style="list-style-type: none"> - Percabangan Bisa Saja kosong. - Percabangan sangat tidak signifikan.
<i>K-Nearest Neighbor</i>	<ul style="list-style-type: none"> - Tangguh terhadap training data yang <i>noisy</i> dan efektif apabila data latihnya besar 	<ul style="list-style-type: none"> - Perlu menentukan nilai parameter - Pembelajaran untuk jarak tidak jelas mengenai jenis jarak apa dan atribut mana yang harus digunakan. - Biaya komputasi tinggi

Penelitian terkait perbandingan beberapa metode klasifikasi lain yaitu oleh Putri et al. (2020) tujuan penelitiannya ini adalah memperoleh metode yang paling baik dan optimal pada klasifikasi *data mining* dengan metode *Naïve Bayes*, *Logistic Regression* serta *Support Vector Machine* diukur menggunakan *confusion matrix* dan *K-fold cross validation*. Dengan dataset pasien penyakit jantung yang di ambil dari *Cleveland Clinic Foundation* sebanyak 303 data pasien dengan 165 pasien terdeteksi sakit dan 138 pasien sehat, serta 14 atribut yang digunakan sebagai diagnosis dalam proses klasifikasinya membagi data menjadi data training sebesar 80% dan testing 20%, setelah proses analisis dari beberapa metode yang telah

dilakukan didapat hasil metode *Naïve Bayes* menghasilkan akurasi 87%, *Logistic Regression* menghasilkan akurasi 82% dan metode *Support Vector Machine* menghasilkan akurasi sebesar 85%, dengan pengukuran menggunakan *confusion matrix*. Sedangkan pengukuran menggunakan *K-fold cross-validation* didapatkan hasil metode *Naïve Bayes* menghasilkan nilai 0.805, *Logistic Regression* menghasilkan nilai 0.835 dan metode *Support Vector Machine* menghasilkan nilai sebesar 0,838. Dengan kesimpulan saat pengujian menggunakan *confusion matrix* dapat dimenangkan oleh metode *Naïve Bayes* dikarenakan sangat bergantung pada keseluruhan data pelatihan yang dihasilkan. sedangkan untuk pengujian menggunakan *K-fold cross-validation* dimenangkan *Support Vector Machine* dikarenakan metode ini hanya melihat sebagian data saja maka nilai akurasi yang dihasilkan menjadi lebih tinggi, dengan kata lain dengan metode berbeda maka akan menghasilkan nilai akurasi yang berbeda pula begitupun dengan teknik yang diterapkan untuk pengukurannya.

Pada penelitian selanjutnya metode *Naïve Bayes* dibandingkan dengan metode *K-Nearest Neighbor* pada proses klasifikasi pada *data mining* serta masih dengan pengukuran menggunakan *confusion matrix* guna menemukan metode yang terbaik. Pada penelitian yang dilakukan oleh Indriani (2020) dengan membandingkan kinerja metode *Naïve Bayes* dan metode *K-Nearest Neighbor*, data yang digunakan adalah data forum yang jumlahnya 21 data dengan 15 data uji. Kemudian dilakukan proses penelitian klasifikasi, dari kedua metode penelitian ini diperoleh nilai akurasi untuk hasil pengukuran dengan menggunakan dasar pengukuran *confusion matrix* metode *Naïve Bayes* menghasilkan nilai akurasi sebesar 80% dan metode *K-Nearest Neighbor* memperoleh nilai akurasi 73%.

Dimana telah ditarik kesimpulan untuk hasilnya bahwa metode *Naïve Bayes* lebih mengarah pada probabilitas tingkat kemunculan kata sedangkan untuk hasil metode *K-Nearest Neighbor* yaitu dengan melihat kedekatan jarak antar data latih terhadap data uji, dengan kesimpulan terakhir bahwa pada penelitian ini metode *Naïve Bayes* lebih unggul dibandingkan metode *K-Nearest Neighbor*.

Setelah melihat hasil dari beberapa penelitian sebelumnya dan dengan melakukan beberapa perbandingan metode, peneliti dapat lebih mempersempit kesimpulan hasil metode *Naïve Bayes* dan *Support Vector Machine* menjadi pilihan dua metode terbaik klasifikasi pada pemrosesan *data mining* berdasarkan pola data dan teknik yang digunakan pada masing-masing metode berkaitan dengan hal tersebut, pada penelitian kali ini dengan membandingkan kedua metode tersebut yang dilakukan oleh Fibrianda & Bhawiyuga (2018) juga melakukan analisis setimen dengan data berbentuk teks pada *machine learning*, dengan tujuan mencari perbandingan terbaik dari metode *Naïve Bayes* dan *Support Vector Machine* dengan karnel (*linear*, *polynomial*, dan *sigmoid*) dengan data set yang digunakan adalah data dari ISCX *tesbed* tanggal 24 juni 2012 serta proses analisis yang telah dilakukannya untuk mendapatkan hasil terbaik dari salah satu metode klasifikasi dengan pengukuran menggunakan *accuracy*, *precision*, *recall*, dan *f1-score* dengan beberapa tahap dengan penjelasan berikut:

1. Hasil tahapan klasifikasi menggunakan metode *behavior based*, dari model yang sudah dibangun diuji menggunakan data *testing* yang menghasilkan sebuah output untuk melihat akurasi *traffic*.
2. Pengolahan dataset ISCX 2012 dengan mengubah format *xml* menjadi *csv* serta mengubah data *string* menjadi *integer* dan proses *cleaning*.

3. Fitur yang digunakan adalah *totalSourceBytes*, *totalDestinationPacket*, *totalSourcePacket*, *directionSource* *TCPflagsDescription*, *protocolName*, *sourcePort*, *Destination*, *destinationPort*, *startDateTime* dan *stopDateTime*.
4. Performa *confusion matrix* yang dihasilkan dari *Classifier Naïve Bayes*, *SVM Linear*, *Polynomial* dan *Sigmoid* sebesar 85%, 99%, 99% dan 99%
5. Performa dari kurva ROC adalah *Classifier Naïve Bayes*, *SVMLinear*, *Polynomial* dan *Sigmoid* sebesar AUC 0.5, 0.75, 0.33 dan 0.5.

Pada penelitian yang telah dilakukan kali ini oleh Rohanah (2021) analisis sentiment berupa data berbentuk teks pada *data mining* menggunakan kedua metode *Naïve Bayes* dan *Support Vector Machine* menjadi suatu pertimbangan yang sangat sulit untuk memilih salah satunya, tidak hanya melihat pada nilai hasil akurasi saja tetapi pada bagaimana karakteristik data serta tujuan yang akan dicapai pada analisis yang dilakukan yang juga sangat mempengaruhi dari hasil masing-masing metode. Pada Penelitian ini dengan tujuan membandingkan kinerja metode *Naïve Bayes* dan *Support Vector Machine* dan mengaplikasikan tahapan penelitian *Cross Industry Standard Process for Data Mining* CRISP-DM, dengan menganalisis tingkat kepuasan layanan pelanggan IndiHome pada data ulasan pelanggan pada aplikasi media sosial Twitter, jumlah data 1000 *tweet* menggunakan *tools RapidMiner* dan *library*, Dengan hasil visualisasi data disajikan dalam bentuk *word cloud* yang dikategorikan dalam opini positif dan negatif. Diperoleh pada penerapan algoritma *Support Vector Machine* dengan penggunaan karnel *linear* dapat menghasilkan nilai *accuracy*, *precision*, *recall* dan *AUC* lebih baik dibandingkan dengan algoritma *Naïve Bayes* dengan perolehan hasil nilai *accuracy* 82,11%, *precision* 76,44%, *recall* 88,01%, dan nilai *AUC* 0,909.

2.2 Feature Extraction

Terkait paparan materi studi pustaka kali ini akan membahas permasalahan dan penanganan yang terjadi pada proses pengolahan *data mining* pada analisis sentimen yang tepatnya di level *feature extraction*, berlandaskan penelitian terdahulu. Seperti penelitian oleh Fransiska & Gufroni (2020) pada penelitiannya dengan menganalisis sentimen pelayanan provider by.U, dengan data yang didapat melalui *google play store* menggunakan bahasa pemrograman python dan teknik yang diterapkan pada *feature extraction* nya adalah *Term Frequency-Inverse Document Frequency* (TF-IDF) yang akan diterapkan, guna menangani permasalahan pada salah satu karakteristik data yang terdapat pada *big data* yaitu data *variety* sangat acak/beragam serta meningkatkan akurasi model analisis yang dibuat. Dengan penggunaan teknik tersebut menghasilkan akurasi dengan rata-rata 84,7%, presisi 84,9%, recall 84,7%, dan fmeasure 84,8%. Hasil akurasi tertinggi pada fold 2, 86,1%. Pengaruh TF-IDF pada pengukuran kinerja model tidak begitu besar tetapi lebih baik, dikarenakan teknik ini dapat menghilangkan kategori pada tiap dokumen dengan cara kerja TF-IDF adalah melakukan pembobotan pada tiap-tiap dokumen dan memberikan bobot tinggi pada *term* yang jarang muncul.

Selanjutnya penelitian yang dilakukan oleh Putri & Hendrowati (2018) dalam jurnalnya dapat dijelaskan hasil dari penelitian tersebut bahwa teknik *Bag of Word* (BOW) merupakan sebuah teks yang berupa kalimat ataupun dokumen diwakili sebagai kantung (*bag*) dengan multiset dari kata-kata yang terkandung di dalamnya dengan cara kerja mengekstrak kalimat menjadi angka berbentuk vektor dengan berbagai perhitungan yang diterapkan pada implementasinya, tanpa memandang urutan kata dan tata bahasa namun masih tetap mempertahankan keberagamannya.

Studi kasus pada penelitian ini adalah dengan menganalisis sentimen dari masyarakat terhadap calon gubernur DKI periode 2016-2021 dengan tujuan analisis sentimen menggunakan *machine learning*. Hasil eksperimen setelah dilakukan proses pengambilan data melalui API Twitter, serta proses preprocessing. Eksperimen ini dimulai dengan mengimpor data *stopwords* bahasa Indonesia ke dalam *tools* pemrograman. Selanjutnya setiap kata disimpan sebagai vektor yang akan dicocokkan dengan dokumen teks *tweet* yang sudah dibersihkan. Setelah dilakukan pengolahan dan analisis terhadap data, sesuai dengan tujuan penelitian, maka dapat disimpulkan bahwa belum ditemukan informasi baru yang terkait dengan hasil pengelompokan dan perhitungan yang disajikan. Model BOW yang digunakan dalam penelitian ini merupakan model sederhana yang cukup efektif untuk menemukan kata (*terms*) paling sering dicuitkan terhadap sebuah entitas.

Pada penelitian kali ini dilakukan oleh Kurniawan & Maharani (2020) Penelitian ini bertujuan memahami sentimen pengguna Indonesia terhadap suatu topik yang dibahas di Twitter dengan menggunakan teknik *feature extraction* Word2Vec. Teknik ini didasarkan pada ide *deep learning* dimana kata direpresentasikan kedalam nilai-nilai angka berupa vektor dengan cara kerja membangun kosakata dari data teks dan kemudian mempelajari representasi vector dari kumpulan kata tersebut agar dapat diproses dan dipahami oleh *machine*. Kemudian kesimpulan dari penelitian ini adalah perbedaan jenis arsitektur model dan ukuran dimensi Word2vec mempengaruhi hasil klasifikasi model *skip-gram* 100 dimensi memberikan hasil klasifikasi paling baik dengan tingkat presisi sebesar 64,4%, *recall* sebesar 58%, dan *f-score* sebesar 61,1%. Hasil klasifikasi paling baik didapatkan ketika menggunakan data latih sebanyak 1010.

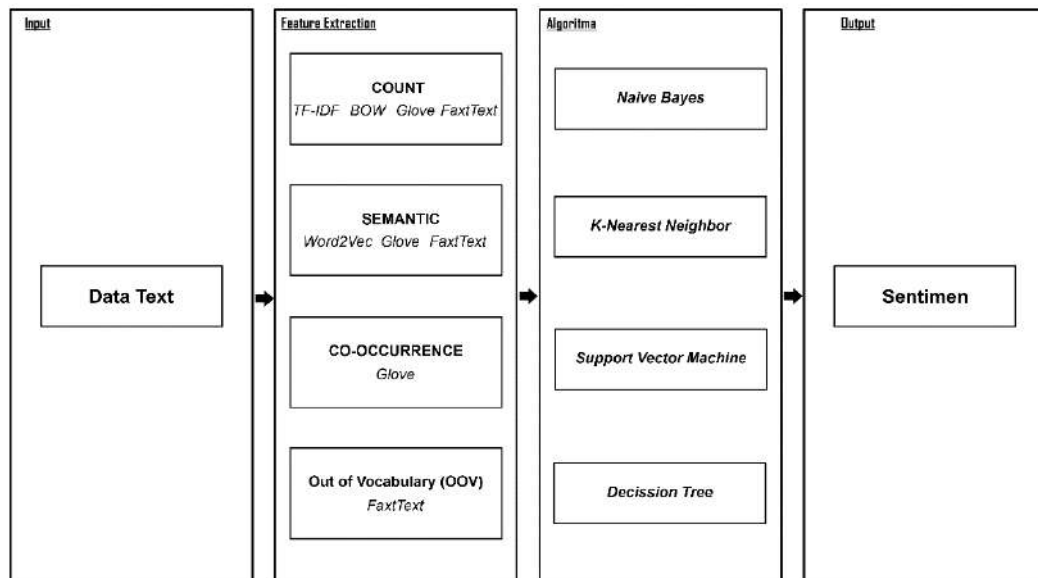
Penelitian selanjutnya oleh Faiq et al. (2022) pada penelitian ini melakukan analisis sentimen pada *feature extraction* dengan menerapkan teknik *GloVe* dalam mengekstraksi data berbentuk teks berupa data kebijakan pemerintah yang bersumber dari aplikasi media sosial Twitter. Menurutnya *GloVe* adalah sebuah teknik *feature extraction* atau teknik yang merepresentasikan teks kedalam vektor berupa angka, dengan tujuan *GloVe* ini adalah dengan mendapatkan hubungan semantic antar kata dari matriks kemunculan (*co-occurrence*) dan teknik ini adalah termasuk pengembangan dari teknik Word2Vec dengan beberapa pengembangan yaitu Word2vec menggunakan model *feed forward neural network* sehingga sering disebut sebagai *neural word embeddings* (NLP), sedangkan *GloVe* menggunakan model *log-bilinear* atau secara sederhana dapat disebut sebagai model berbasis hitungan. Kemudian menurutnya ada beberapa kelebihan teknik *feature extraction GloVe ini* meliputi kemampuannya dalam menangkap hubungan semantik antarkata, representasi dimensi yang lebih rendah, ketersediaan model *pre-trained* pada korpus besar dan pemahaman kontekstual kata yang baik juga dapat merepresentasikan kemiripan semantik antar kata, memudahkan identifikasi hubungan makna kata. Walaupun *GloVe* memiliki banyak kelebihan, kinerjanya dapat tergantung pada tugas dan karakteristik dataset yang digunakan. Meskipun begitu, metode ini tetap menjadi pilihan yang populer dalam pemrosesan bahasa alami atau pengestarakan kata menjadi angka berupa vektor. Hasil akhir dari penelitian dengan menggunakan *feature extraction GloVe* terbukti dapat meningkatkan akurasi sebesar 4,77%. Dari pengujian yang telah dilakukan didapatkan juga hasil paling optimal untuk akurasi pada fitur top 1 dengan nilai *accuracy* sebesar 79,52% dan nilai *f1-Score* sebesar 0,7942.

Penelitian kali ini yang dilakukan oleh Dikih Arif Wibowo (2020) pada penelitiannya mengenai penanganan pro dan kontra terhadap vaksinasi di Indonesia dengan langkah melakukan analisis sentimen untuk mengetahui pendapat atau pandangan masyarakat terhadap vaksin. Menurutnya berbagai penelitian telah dilakukan dan *Word2vec* sebagai *feature extraction* yang memiliki keunggulan dapat melihat hubungan semantic antar kata namun memiliki kelemahan dalam merepresentasikan kata yang tidak ada didalam training kamus dengan masalah yang dihadapi paada penelitian ini peneliti menerapkan penelitian menggunakan teknik *FastText* yang diangkat sebagai terobosan pemecahan masalah yang sudah dihadapi, dengan cara kerja *Word2vec* merepresentasikan kata sebagai sekumpulan *n-gram* berbentuk karakter. Kemudian untuk permasalahan lain seperti kata yang tidak ada dalam proses training yang telah dilakukan atau dalam korpus dapat ditangani menggunakan teknik *FastText*. Dengan proses pengumpulan data yang digunakan adalah dengan mengambil data dari aplikasi media social bernama Twitter berjumlah 832 data yang terdiri dari 672 data yang akan digunakan sebagai data latih, dan 160 data yang akan digunakan sebagai data uji. Untuk dapat mengenali pola data, digunakan metode klasifikasi SVM (*Support Vector Machine*). Target klasifikasi analisis sentimen ini terdiri dari kelas positif, negatif, dan netral. Berdasarkan penelitian yang telah dilakukan teknik *FastText* mendapatkan hasil akurasi sebesar 88.1% dengan arsitektur *skip-gram*, sementara untuk perbandingan teknik *Word2vec* mendapatkan akurasi sebesar 76.8% dengan arsitektur CBOW. Dengan kesimpulan pemakaian teknik *feature extraction* yaitu *FastText* dapat diandalkan untuk permasalahan *out of vocabulary* (OOV), juga *FastText* unggul saat melakukan pemrosesan kata tidak baku dan salah penulisan *typo*.

Selanjutnya penelitian yang dilakukan oleh Nurdin et al. (2020) pada penelitian ini mengangkat permasalahan tentang karakteristik teks yang tidak terstruktur dengan dihadapkan pada banyaknya bahasa serta gaya penulisan yang berbeda beda setiap orang, menjadi tantangan dalam ekstraksi fitur pada bidang pemrosesan teks. Penelitian ini mempunyai tujuan membandingkan kinerja pada teknik *feature extraction* yaitu *Word2Vec*, *GloVe* dan *FastText* yang diklasifikasikan menggunakan *algoritma Convolutional Neural Network*. Dengan melakukan proses atau alur penelitian sesuai dengan alur klasifikasi yaitu mulai dari pengambilan data, pembersihan data, modeling data menggunakan algoritma sampai dengan akhir yaitu validasi dan evaluasi model yang telah dibuat, dari ketiga teknik yang telah dipilih didasari karena dapat menangkap makna secara semantik, sintatik, dan berurutan bahkan konteks di sekitar kata juga dapat mempengaruhi hasil dari pemrosesan yang telah dilakukan, kemudian jika dibandingkan teknik *word embedding* tradisional seperti *TF-IDF* dan *BOW* masih lebih baik dari ketiga teknik tersebut. Dengan proses analisis sentimen menggunakan *machine learning* data yang digunakan dalam penelitian ini berupa berita dari dataset *20 newsgroup* dan *Reuters Newswire* yang diukur menggunakan *F-Measure*. Mendapatkan hasil kesimpulan performa terbaik adalah *FastText* lebih unggul dibanding kedua metode *Word Embedding* lainnya yaitu *Word2Vec* dan *GloVe* dengan hasil nilai akurasi didapat *F-Measure* sebesar 0.979 untuk dataset *20 newsgroup* dan 0.715 untuk *reuters*. Namun, hasil perbedaan kinerja yang tidak begitu signifikan antara ketiga teknik *feature extraction* tersebut menunjukkan bahwa ketiganya ini memiliki kinerja yang sama-sama kompetitif. Penggunaannya sangat bergantung pada dataset yang digunakan dan permasalahan yang ingin diselesaikan.

2.3 Kerangka Teori

Penyusunan kerangka teori ini beracuan dari beberapa penelitian terdahulu yang telah dipaparkan, terkait permasalahan maupun penanganan pada proses klasifikasi teks menggunakan *machine learning* dengan tujuan analisis sentimen.



Gambar 2. 6 Kerangka teori

Pada Gambar 2.6 kerangka teori yang digambarkan dengan penjelasan terbagi menjadi dua bagian yaitu pencarian beberapa teknik *feature extraction* terbaik dan pencarian metode analisis terbaik dengan paparan sebagai berikut:

- a. Pada pembahasan kali ini adalah merangkum semua dari proses analisis sentimen dengan data awal adalah data teks dengan semua tujuan akhir adalah sentimen untuk mencari dan menerapkan beberapa teknik yang terbaik guna melakukan penyelesaian dalam permasalahan pada level *feature extraction* seperti yang sudah dipaparkan pada bahasan penelitian-penelitian terkait sebelumnya, berikut adalah tabel beberapa teknik-teknik yang digunakan peneliti terdahulu yang dijelaskan beserta dengan hasil kesimpulannya:

Tabel 2. 2 Perbandingan teknik *feature extraction*

No	Judul & Penulis	Dataset	Teknik	Hasil
1	<i>Sentiment Analysis Provide by.U on Google Play Store Riviws With TF-IDF and Support Vector Machine (SVM) Method</i> (Fransiska & Gufroni, 2020).	Riview Google Play Store	TF-IDF	Pengaruh TF-IDF pada pengukuran kinerja model tidak begitu bagus, tetapi masih lebih baik. Dengan rata-rata accuracy 84.7%.
2	<i>Penggalian Teks Dengan Model Bag of Word Terhadap Data Twitter</i> (Putri & Hendrowati, 2020).	Data Teks Twitter	Bag of Words	Efektif untuk menemukan kata-kata (terms) yang paling sering muncul dan terdaftar pada korpus yang disediakan
3	<i>Analisis Sentimen Twitter Bahasa Indonesia Dengan Word2Vec</i> (Kurniawan & Maharani (2020).	Data Teks Twitter	Word2Vec	Word2Vec memiliki keunggulan dapat melihat hubungan semantik antar kata atau hubungan yang memiliki makna kata sama
4	<i>Analisis Sentimen Terhadap Kebijakan Pemerintah dengan Feature Expansion Metode Glove Pada Media Sosial Twitter</i> (Faiq et al, 2022).	Data Teks Twitter	GloVe	Penggunaan feature expansion GloVe memiliki akurasi cukup baik dan dapat meningkatkan akurasi sebesar 4,77% dari baseline dengan akurasi optimal sebesar 79,52%
5	<i>Analisis Sentimen Tweet Bahasa Indonesia Tentang Vaksin Covid-19 Menggunakan FastText Embedung dan SVM</i> (Wibowo & Musdholifah, 2021).	Data Teks Twitter	FastText	FastText unggul pada permasalahan out of vocabulary, serta juga unggul saat berhubungan dengan kata tidak baku dan kesalahan penulisan
6	<i>Perbandingan Kinerja WordEmbeding Word2Vec, GloVe dan FastText Pada Klasifikasi Teks</i> (Nurdin et al. 2020).	Dataset 20 newsgroup dan Reuters Newswire	Word2Vec, GloVe dan FastText	Kinerja terbaik diperoleh oleh teknik FastText yang mampu dan dapat merepresentasikan vektor dari kata yang tidak ada dalam kamus (out of vocabulary). Namun, GloVe juga memiliki keunggulan pada cepatnya proses kerja <i>machine</i> .

Mengacu pada Tabel 2.2 tentang perbandingan dan penerapan teknik-teknik pada permasalahan *feature extraction* yang digunakan oleh peneliti terdahulu dan dipaparkan pada tabel diatas didapat kesimpulan dua teknik yang paling optimal dalam segala nilai aspek dan sangat direkomendasikan serta memiliki kekurangan dan kelebihan dari masing-masing teknik. Kedua teknik tersebut adalah *GloVe* dan *FastText*, dengan beracuan pada hasil beberapa perbandingan dan keunggulan-keunggulan yang ada dari kedua teknik tersebut serta terdapat beberapa kesamaan hadalam hal data dan tujuan penelitian yang akan dilakukan, peneliti menerapkan dan membandingkan kedua teknik guna mendapatkan hasil yang paling optimal dari salahsatunya dengan tujuan analisis sentimen menggunakan data teks bersumber dari media sosial Twitter.

- b. Setelah mendapatkan beberapa hasil terbaik dan menentukan dua teknik yang dipilih untuk dijadikan perbandingan penerapan teknik dalam penyelesaian *feature extraction*, selanjutnya bagian terpenting lagi adalah mencari metode yang akan diterapkan dari hasil beberapa perbandingan penelitian terdahulu, terdapat beberapa metode yang digunakan dalam penerapan proses analisis sentimen dengan data berbentuk teks dengan tujuan masih sama yaitu analisis sentimen pada penelitian terkait. Peran algoritma atau metode pada proses analisis sentimen pada *machine learning* ini adalah mempelajari pola dan konteks dari data teks sehingga dapat secara otomatis mengidentifikasi dan mengklasifikasikan data dengan beberapa persamaan yang telah diterapkan dan dipatenkan pada masing-masing metode. Melihat sangat pentingnya implementasi metode pada analisis sentimen guna mendapatkan hasil yang optimal, peneliti akan memilih metode dari beberapa perbandingan yang ada:

Tabel 2. 3 Perbandingan metode klasifikasi

No	Judul & Penulis	Dataset	Metode	Hasil
1	<i>Fundamental of Python for Machine Learning</i> (Wahyono, 2018)	Jenis Metode Klasifikasi	<ul style="list-style-type: none"> - K-Nearest Neighbor - Decision Tree - Naïve Bayes - Support Vector Machine 	Adalah beberapa rekomendasi dari metode-metode klasifikasi yang umum digunakan.
2	<i>Metode – Metode Klasifikasi</i> (Wibawa et al, 2018)	Kekurangan dan Kelebihan Metode	<ul style="list-style-type: none"> - Naïve Bayes - Support Vector Machine - Decision Tree - K-Nearest Neighbor 	Support Vector Machine (SVM) memiliki poin kelebihan terbanyak
3	<i>Comparasion of Data Mining Classification Methods to Detect Heard Disease</i> (Putri et al, 2020)	Pasien Penyakit Jantung	<ul style="list-style-type: none"> - Naïve Bayes - Support Vector Machine 	Pengukuran akurasi dilakukan menggunakan Confusion matrix dimenangkan Naïve Bayes sedangkan K-fold cross-validation dimenangkan <i>Support Vector Machine</i> (SVM)
4	<i>Analisa Perbandingan Metode Naïve Bayes Classifier dan K-Nearest Neighbor Terhadap Klasifikasi Data</i> (Indriani, 2020)	Sumber Data Forum diskusi	<ul style="list-style-type: none"> - Naïve Bayes - K-Nearest Neighbor 	Dimenangkan metode Naïve Bayes dengan akurasi (Naïve Bayes 80% vs KNN 73%)
5	<i>Analisis perbandingan Akurasi Deteksi Serangan pada Jaringan Komputer Dengan Metode Naïve Bayes dan Support Vector Machine (SVM)</i> (Fibrianda & Bhawiyuga, 2018)	Dataset ISCX 2012	<ul style="list-style-type: none"> - Naïve Bayes - Support Vector Machine 	Dimenangkan metode Support Vector Machine (SVM) dengan karnel <i>polynomial</i>
6	<i>Perbandingan Naïve Bayes dan Support Vector Machine Untuk Klasifikasi Ulasan Pelanggan Indihome</i> (Rohonah, 2021)	Data Teks Twitter	<ul style="list-style-type: none"> - Naïve Bayes - Support Vector Machine 	Dengan hasil Dimenangkan oleh metode <i>Support Vector Machine</i> (SVM) dengan mendapatkan nilai akurasi 82.11%.

Mengacu pada Tabel 2.3 tentang perbandingan dan penerapan metode-metode klasifikasi teks untuk analisis sentimen pada *machine learning* seperti metode *K-Nearest Neighbor* (KNN), *Decision Tree*, *Naïve Bayes* dan *Support Vector Machine* (SVM) dengan dataset berbentuk teks dan tujuan yang sama yaitu menghasilkan klasifikasi sentimen yang telah dilakukan oleh peneliti terdahulu terdapat beberapa hasil dengan kelebihan dan kekurangan pada masing metode yang juga sangat berpengaruh pada beberapa aspek kecil lain seperti jumlah dataset yang digunakan sebagai penelitian, dll. Kesimpulan peneliti adalah dengan beracuan pada banyak aspek dan hasil serta kesamaan dataset yang digunakan pada penelitian ini memilih dan menerapkan metode yang paling unggul yaitu *Support Vector Machine* (SVM) dengan menggunakan kernel *polynomial* pada proses klasifikasi analisis sentimen yang akan dilakukan serta ditinjau berdasarkan hasil keunggulan metode *Support Vector Machine* (SVM) mempunyai hasil akurasi dominan yang lebih unggul dibandingkan dengan metode-metode pembandingan yaitu *K-Nearest Neighbor*, *Decision Tree* dan *Naïve Bayes*. Juga proses klasifikasi yang dihasilkan dan pengukuran modeling yang dilakukan serta beberapa keunggulan-keunggulan lain yang dimilikinya seperti yang telah dipaparkan pada Tabel 2.3 secara jelas dan rinci, dari kesamaan jenis dataset yaitu dataset dengan data berbentuk teks yang bersumber dari media sosial dengan karakteristik data yang acak dan tidak beraturan secara suku kata, begitu pula dengan alur proses penelitian maupun teknik pengujian yang digunakan sebagai dasar dari hasil kesimpulan pemilihan yang dilakukan dengan kesamaan alur mulai dari cara pengambilan data awal sampai dengan proses pengujian dan evaluasi hasil yang didapatkan.

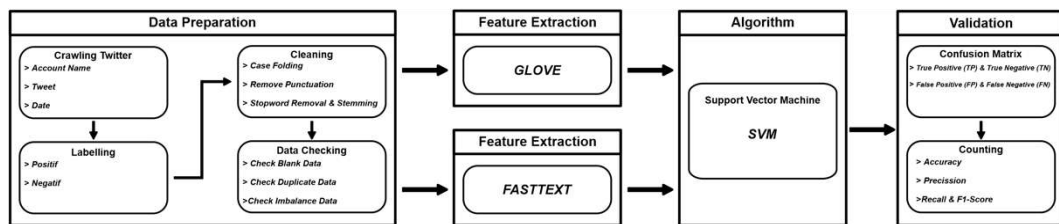
BAB III

KONSEPTUAL MODEL DAN DATA PREPARATION

Bagian bab ini membahas representasi dari sebuah sistem secara rinci pada setiap langkah penelitian serta membahas proses persiapan data mentah hingga siap diproses untuk dilakukan klasifikasi teks menggunakan *machine learning*.

3.1 Konseptual Model

Berikut merupakan representasi alur proses penelitian klasifikasi teks untuk analisis sentimen menggunakan *machine learning* secara utuh diterapkan pada penelitian kali ini, yang dipaparkan pada Gambar 3.1 dibawah disertai penjelasan singkat pada setiap alur berdasarkan cara kerja serta proses didalamnya.



Gambar 3. 1 Konseptual Model

Langkah pertama dari penelitian ini adalah *data preparation* atau mempersiapkan data sebelum diproses yang terdiri mulai dari *crawling* atau penambangan data mentah berbentuk teks yang bersumber dari aplikasi Twitter berisi nama pengguna, isi komentar dan tanggal unggah dengan jumlah data yang akan diambil yaitu sebanyak 1000 data cuitan. Kemudian *labelling* atau pelabelan data yang akan dilakukan oleh pihak terkait yaitu karyawan JNT Express dan diproses sebagai syarat untuk pengolahan data *supervised* serta proses terakhir adalah *cleaning*, data tersebut dibersihkan dari fitur-fitur yang tidak penting yang bisa mempengaruhi kinerja dari *machine* jika diproses secara langsung dan setelah semua siap akan dilanjutkan kedalam pengolahan kata atau *feature extraction*.

Langkah kedua adalah *feature extraction*, dalam proses ini data yang sudah melewati langkah pertama atau *data preparation* maka data sudah memenuhi syarat yaitu bersih dan berlabel yang akan diolah dengan cara kerja menggambarkan kata kedalam bentuk angka berupa vektor, penerapan pada proses klasifikasi analisis sentimen ini dengan tujuan agar komputer dapat menangkap dan memproses data berupa teks yang akan diolah pada proses selanjutnya. Tidak hanya dengan merepresentasikan kata kedalam angka saja, banyak aspek yang perlu diperhatikan dalam pemrosesan pada tahap *feature extraction* ini seperti hubungan antar kata, kesamaan kata dengan satu makna, dan lain sebagainya. Dikarenakan banyaknya jenis teks berupa bahasa yang dimiliki manusia maka sangat diperlukan teknik yang paling sesuai untuk melakukan pemrosesan pada tahap ini. Dengan paparan yang sudah dijelaskan pada bab sebelumnya peneliti membandingkan dengan kedua teknik terbaik pada proses *feature extraction* yang teknik *GloVe* dan teknik *FastText*.

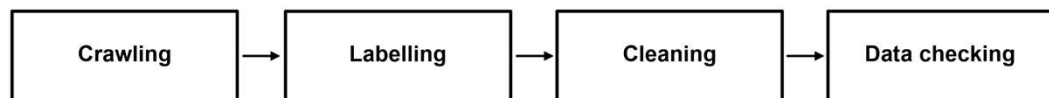
Langkah ketiga setelah data melewati proses *cleaning* dan proses *feature extraction*, saat ini data menjadi bentuk numerik berupa vektor lalu pada tahap inilah jug dilakukan proses modeling atau penerapan metode yang telah dipilih dengan mempelajari kata yang sudah di training sebelumnya atau proses utama dari klasifikasi teks dengan menerapkan metode *Support Vector Machine* (SVM). Algoritma pembelajaran ini pertama kali dikembangkan oleh Boser, Guyon dan Vapnik pada tahun 1992 yang merupakan system pembelajaran dengan menggunakan hipotesis berupa fungsi-fungsi linier didalam sebuah fitur yang memiliki dimensi tinggi dan dilatih menggunakan teori optimasi, yang dilakukan dengan ketelitian yang tinggi untuk mendapatkan model hasil akurat serta optimal.

Pada dasarnya konsep dan cara kerja dari metode *Support Vector Machine* (SVM) adalah berusaha mencari dan menemukan fungsi garis pemisah (*hyperplane*) yang terbaik diantara beberapa fungsi. Formulasi algoritma ini dapat dibedakan menjadi beberapa fungsi atau kernal yang sering digunakan yaitu *linear*, *polynomial*, *radial basis function* dan *sigmoid*. terdapat 4 fungsi dan masing-masing memiliki garis pemisah yang berbeda-beda pada setiap fungsinya. Metode *Support Vector Machine* (SVM) mencari dan memilih dari salah-satu garis/*hyperplane* yang paling baik atau paling optimal dalam melakukan klasifikasi dan pemilihan tersebut sangat bergantung pada jenis data dengan tujuan dapat memisahkan antar kelas yang berbeda dengan sempurna.

Langkah keempat atau yang terakhir untuk proses analisis sentimen menggunakan *machine learning* ini adalah melakukan pengukuran dan evaluasi model yang telah dibuat sebelumnya menggunakan acuan hasil pengukuran *confusion matrix* dalam mengevaluasi performa algoritma *machine learning* dengan melihat hasil *true positive* (TP), *true negative* (TN), *false positive* (FP) dan *false negative* (FN). Setelah melakukan proses *evaluation* pada *confusion matrix* acuan yang digunakan selanjutnya adalah pengukuran nilai *accuracy* yaitu persentase prediksi benar dari *true positive* dan *true negative*, nilai *precision* yaitu persentase nilai *true positive* dari seluruh nilai positif yang diprediksi, nilai persentase prediksi positif dibandingkan dengan *true positive recall* dan nilai *F1 score* yaitu perbandingan rata-rata *precision* dan *recall*. Kedua teknik yang diterapkan pada *evaluation* ini bekerja dengan merepresentasikan prediksi dan kondisi dari data *training* atau data pelatihan dengan data *testing* atau data percobaan yang sudah di pisahkan berdasarkan persentase umum yaitu 80 : 20%.

3.2 Data Preparation

Serangkaian proses mempersiapkan data awal atau data primer pada analisis sentimen dengan karakteristik data berbagai atribut yang tidak diperlukan serta tidak terstruktur diubah menjadi data yang terstruktur dengan beberapa langkah yang harus dilakukan agar dapat diproses dan olah oleh komputer. Tahapan ini merupakan tahap paling awal yang harus dilakukan untuk proses analisis sentimen menggunakan *machine learning*.



Gambar 3. 2 Alur data *preparation*

Pada Gambar 3.2 diatas merupakan langkah secara berurutan untuk menghasilkan data yang siap diproses dan dilakukan analisis, mulai dari *crawling* atau penambahan data mentah berupa cuitan teks kalimat dari Twitter untuk disimpan kemudian *labelling* adalah pelabelan data oleh pihak yang terkait dalam hal ini adalah karyawan J&T Express yang akan diproses sebagai syarat untuk data supervised dan kemudian *cleaning* setelah mendapatkan data dan melabelinya data tersebut dibersihkan dari fitur-fitur yang tidak penting yang bisa mempengaruhi kinerja dari *machine*. Dalam rangkaian proses data *preparation* tersebut penambahan langkah terakhir yaitu data *checking* dengan data yang sudah melewati proses *labelling* dan *cleaning* harus di cek terlebih dahulu seperti keseimbangan antar labelnya, adanya data kosong dan data duplikat sebelum dilanjutkan ke pemrosesan, agar mendapatkan data yang benar benar baik untuk diproses. Berikut merupakan penjelasan detail dari setiap langkah pada proses data *preparation*.

3.2.1 Crawling

Pengambilan data atau *crawling* adalah proses menggali lebih jauh dan mengimpor informasi atau data yang telah ditemukan kedalam file lokal komputer pribadi, tahapan paling awal yang harus dilakukan cara ini menggunakan bantuan *google colab* dengan bahasa pemrograman *python* dari pengguna aplikasi Twitter berupa *tweet* dengan hastag (#J&T.id) yang berfungsi sebagai penanda untuk *tweet* yang saling berhubungan dan menyampaikan ekspresi terkait dengan apa yang akan diteliti yaitu jasa penyedia layanan antar barang J&T Express dengan (.id) sebagai kode untuk wilayah pengambilan yaitu indonesia, karakteristik data berupa teks yang masih mentah dan berisikan angka, simbol, emotion serta huruf di dapatkan dengan sistem *Application Programming Interface (API)* yang secara legal disediakan oleh pihak Twitter. Pengambilan data berdasarkan postingan yang akan diambil adalah mulai dari 11 April 2020 sampai dengan 23 April 2020, sejak diterapkannya awal PSBB (Pembatasan Sosial Berskala Besar) di Indonesia. Batasan pengambilan hasil data mencakup 1000 entri yang berisi variabel nama akun (*action name*), isi teks unggahan (*tweet*), serta waktu unggah (*date*). Proses ini dilakukan dengan menggunakan *source code* yang telah disusun sebelumnya, dengan menghasilkan *source code* dan informasi sebagai berikut:

```

consumer_key = 'i9snFcEsKXI14Z99twYpIpY61'
consumer_secret = "GWwNhozndzdeWaHoyN4iEYPnoDemiWAc8GAHPmAK2"
access_token = "1213346230182395904-0hONPcOlqhatU94rw2EW"
access_token_secret = 'D21JlKqR50At3BM9Jfmqm4b1hls32INVlmy'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

tweets = []
count = 1

```

```

for tweet in tweepy.Cursor(api.search, q='jnt', lang='id',
count='1000', since='2020-04-11', until='2020-04-
23').items(1000):
    count += 1
    try:
        data = [tweet.user.json['screen_name'], tweet.text,
tweet.created_at]
        data = tuple(data)
        tweets.append(data)
    except tweepy.TweepError as e:
        print(e.reason)
        continue
    except StopIteration:
        break

df = pd.DataFrame(tweets, columns = ['Account_Name', 'Tweet',
'Date'])

```

Gambar 3. 3 Source code crawling data Twitter

Source code pada Gambar 3.3 di atas menggunakan library *Tweepy* untuk mengakses API Twitter, mencari dan mengambil 1000 *tweet* dalam bahasa Indonesia yang berdasarkan kata '*jnt*' antara tanggal 11 April 2020 hingga 23 April 2020. Data akun, teks *tweet*, dan tanggal dibaca dan disimpan dalam sebuah *DataFrame* dengan *Pandas* untuk menghasilkan sekumpulan data *excel* yang sudah tersusun berformat *.xlsx* dengan hasil berikut:

Tabel 3. 1 Hasil *crawling* data Twitter

No	Account Name	Tweet	Date
0	@NanaZairel	Paketanku dari tgl 13 STUCK di semarang...□	04/11/2020 13:30
1	@secretleeeey	Harusnya Hari ini sampe @brodi_33	04/11/2020 14:35
...
998	@heydikoShoop	di @jntexpressid , costumer nanyain kejelasan ke kurir malah\diblok. □	04/19/2020 15:40
999	@Mikrokosmos_0613	Coba aja jnt paketku skincare aman sih. Cuma agak lambat www.jnt.co.id	04/19/2020 16:45

3.2.2 Labelling

Dalam hasil data yang sudah di *crawling* atau ditambah pada aplikasi Twitter yang berupa data teks berbentuk kalimat akan diberi label menjadi dua variabel yaitu positif dan negatif. Pada fase ini dilakukan secara manual terkait *tweet* yang bersifat mengarah pada pendapat yang positif atau negatif, dengan dilakukan oleh orang yang berkompeten dibidangnya pada kasus penelitian ini adalah J&T Express maka akan dilakukan oleh orang yang bekerja di bidang tersebut. Kemudian pada pelabelan data ini menggunakan teknik *crowdsourcing*. Adalah suatu teknik melakukan suatu pekerjaan dengan bantuan beberapa orang (lebih dari 1 orang dengan jumlah ganjil) untuk menentukan suatu pencapaian atau keputusan yang valid dari jumlah voting terbanyak antar keputusan, dengan teknik ini peneliti menggunakan tiga orang karyawan yang bekerja pada perusahaan J&T Express untuk melakukan pelabelan data, dengan hasil dari pelabelan yang telah dilakukan oleh ketiga orang tersebut maka akan mendapatkan suatu keputusan dengan jumlah kesamaan hasil label berjumlah dua berbanding satu, dengan hasil akhir ketentuan label yang berjumlah dua persamaan, maka itulah hasil keputusan label yang akan dipakai.

Tabel 3. 2 Hasil labeling menggunakan teknik *crowdsourcing*

No	Tweet	Label 1	Label 2	Label 3	Hasil
0	Paketanku dari tgl 13 STUCK di semarang...☐	Negatif	Negatif	Negatif	Negatif
1	Harusnya Hari ini sampe @brodi 33	Negatif	Negatif	Positif	Negatif
...
998	di @jntexpressid , costumer nanyain kejelasan ke kurir malah\diblok. ☐	Negatif	Negatif	Negatif	Negatif
999	Coba aja jnt paketku skincare aman sih. Cuma agak lambat www.jnt.co.id	Positif	Positif	Negatif	Positif

3.2.3 Cleaning

Dalam *Natural Language Processing* (NLP), sebagian besar teks dan dokumen mengandung banyak kata yang berlebihan seperti *stopwords*, *misspellings*, *slangs*, dan lain-lain. Pada bagian ini, menjelaskan secara singkat beberapa teknik dan metode untuk pembersihan teks dan pra-pemrosesan dokumen teks. Dalam banyak algoritma seperti metode pembelajaran statistik dan probabilistik, noise dan fitur yang tidak perlu dapat mempengaruhi kinerja keseluruhan secara negatif, jadi penghapusan fitur-fitur ini sangatlah penting.

a. Case Folding/Capitalization

Pada suatu kalimat yang terdapat pada cuitan *tweet* biasanya dapat berisi campuran huruf besar/kapital dan huruf kecil beberapa campuran huruf tersebut membentuk kalimat dokumen teks yang dapat mempengaruhi kinerja proses berjalannya *machine learning* jika tidak ditangani, penanganan pada permasalahan ini adalah dengan merubah atau menyamakan semua isi teks tersebut menjadi huruf kecil secara keseluruhan. Pada tahap ini menggunakan teknik *lowercase*, dengan *source code* dan hasil sebagai berikut:

```
def clean_text(text):
    text = text.lower() #lowercase
    return text

clean1 = lambda x: clean_text(x)
df['clean1'] = pd.DataFrame(df['tweet'].apply(clean1))
```

Gambar 3. 4 *Source code case folding*

Source code pada Gambar 3.4 diatas melakukan pembersihan data teks berupa *tweet* dengan mengubah semua hurufnya menjadi huruf kecil atau *lowercase*, dan hasilnya disimpan dalam kolom baru bernama '*clean1*' dalam *DataFrame* 'df' dengan hasil berikut yang diperoleh berikut:

Tabel 3. 3 Hasil *case folding*

No	Tweet	Clean 1
0	Paketanku dari tgl 13 STUCK di semarang...□	paketanku dari tgl 13 stuck di semarang...□
1	Harusnya Hari ini sampe @brodi_33	harusnya hari ini sampe @brodi_33
...
998	di @jntexpressid , costumer nanyain kejelasan ke kurir malah\diblok. □	di @jntexpressid , costumer nanyain kejelasan ke kurir malah\diblok. □
999	Coba aja jnt paketku skincare aman sih. Cuma agak lambat www.jnt.co.id	coba aja jnt paketku skincare aman sih. cuma agak lambat www.jnt.co.id

b. Remove Punctuation

Setelah proses penyamarataan huruf atau *case folding* selesai, dokumen teks yang didapat umumnya masih ada berisi karakter seperti tanda baca atau karakter khusus dan tidak diperlukan seperti url, angka, emotion dan spasi berlebih. *Remove punctuation* ini berperan untuk menghapus sebagian tanda baca yang sangat mengganggu dan memperlambat proses *machine* berjalan, meskipun terkadang tanda baca sebagian juga penting untuk memahami arti dan konteks kalimat dengan *source code* dan hasil sebagai berikut:

```
def clean_text(text):

    text = re.sub('@[^\s]+', '', text)
    text = re.sub('\.[.*?\]', '', text)
    text = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', '',
text)
    text = re.sub('[%s]' % re.escape(string.punctuation),
text)
    text = re.sub('\w*\d\w*', '', text)
    text = re.sub('[\'\""...]', '', text)
    text = re.sub('\n', '', text)
    return text
clean2 = lambda x: clean_text(x)
df['clean2'] = pd.DataFrame(df['tweet'].apply(clean2))
```

Gambar 3. 5 Source code remove punctuation

Source code pada Gambar 3.5 bekerja dengan melakukan pembersihan teks *tweet* dari elemen seperti nama pengguna, tautan, karakter numerik, tanda baca, dan karakter spesial lainnya, lalu menyimpan hasilnya dalam kolom '*clean2*' dalam *DataFrame* '*df*' dengan hasil berikut:

Tabel 3. 4 Hasil *remove punctuation*

No	Clean 1	Clean 2
0	paketanku dari tgl 13 stuck di semarang...□	paketanku dari tgl stuck di semarang
1	harusnya hari ini sampe @brodi_33	harusnya hari ini sampe
...
998	di @jntexpressid , costumer nanyain kejelasan ke kurir malah\diblok. □	di costumer nanyain kejelasan ke kurir malah diblok
999	coba aja jnt paketku skincare aman sih. cuma agak lambat www.jnt.co.id	coba aja jnt paketku skincare aman sih cuma agak lambat

c. Stopword Removal

Langkah selanjutnya adalah *stopword removal*, *Stop-word* sendiri didefinisikan sebagai sekumpulan kata yang tidak berhubungan (*irrelevant*) dengan subyek utama yang dimaksud, meskipun kata tersebut sering muncul didalam data yang digunakan. Kata-kata yang dimaksud biasanya adalah jenis kata penghubung seperti di, ke, yang dan lain sebagainya. Berikut adalah algoritma yang digunakan untuk menghilangkan *stopwords* dari kata kunci pencarian yang dimasukkan oleh pengguna:

- Memasuka *stopword* dari database kedalam *array stoplist* di variabel;
- Memecah variabel string menggunakan fungsi *string split* ke dalam *array*;
- Inisialisasi kata yang sudah didapatkan berupa variabel yang berisikan nilai-nilai *boolean false*;

- Apakah elemen pada array katakunci samadengan elemen pada *stoplist*?
Jika *true*, lakukan langkah 6, jika tidak terpenuhi ubah nilai menjadi *true*;
- Lakukan langkah 4-5 hingga seluruh elemen pada *array stoplist* habis;
- Apakah variabel ketemu bernilai *false*? Jika kondisi terpenuhi, lakukan langkah 9, jika tidak terpenuhi lakukan langkah 10;
- Masukkan elemen array kata kunci yang dipilih ke dalam array hasil;
- Lakukan langkah 4-9 hingga seluruh elemen pada array kata kunci habis.

Dari algoritma di atas, secara umum teknik ini akan menyeleksi dan menghapus kata-kata penghubung yang tidak penting dalam sebuah kalimat atau dokumen, yang jika tidak dilakukan akan menghambat kinerja *machine*.

Berikut merupakan *source code* dan hasil dari proses *stopword removal*:

```
additional = ['jnt', 'rt']
sw = set().union(stopwords.words('indonesian'), additional)
df['clean3'] = pd.DataFrame(df['clean2'].apply(lambda x: '
'.join([word for word in x.split() if word not in (sw)])))
```

Gambar 3. 6 *Source code* *stopword removal*

Source code pada tabel 3.8 menghapus kata-kata penghubung umum yang tidak relevan dari teks *tweet* seperti di, dan, ke, yang dll. Kedalam kolom '*clean2*' disimpan pada '*clean3*' dengan hasil pemrosesan berikut:

Tabel 3. 5 Hasil *stopword removal*

No	Clean 2	Clean 3
0	paketanku dari tgl stuck di semarang	paketanku tgl stuck semarang
1	harusnya hari ini sampe	harusnya hari sampe
...
998	di costumer nanyain kejelasan ke kurir malah diblok	costumer nanyain kejelasan kurir malah diblok
999	coba aja jnt paketku skincare aman sih cuma agak lambat	coba jnt paketku skincare aman cuma lambat

d. Stemming

Langkah yang paling terakhir untuk proses *cleaning* data atau proses pembersihan data ini adalah *stemming*, yang merupakan proses pemotongan imbuhan (*affixes*) pada kata berimbuhan seperti awalan (*prefixes*), akhiran (*suffixes*), sisipan (*infixes*), dan kombinasi (*confixes*) yang dilakukan dengan menggunakan algoritma untuk mengembalikan suatu kata ke bentuk dasarnya. Sebagai contoh kata bersama, kebersamaan, menyamai, akan di *stem* ke *root word* yaitu "sama". Secara umum *stemming* juga bisa dikatakan sebagai proses atau teknik dalam menemukan kata dasar dari suatu kata umum sebelumnya yang sudah banyak menggunakan variasi kata yang sangatlah banyak persamaan antara satu kata dengan kata yang lain, dengan tanpa menghilangkan makna. *Stemming* ini sendiri juga bisa memiliki fungsi untuk menghilangkan variasi-variasi *morfologi* yang melekat pada sebuah kata dengan cara menghilangkan imbuhan-imbuhan pada kata tersebut, sehingga nantinya di dapat suatu kata yang benar-benar sesuai struktur *morfologi* bahasa indonesia yang baik, dengan tujuan pengurangan jumlah huruf dalam kata yang hanya memiliki satu makna guna mempercepat dan mempermudah saat melakukan proses modeling menggunakan *machine learning*, berikut *source code* dan hasil teknik *stemming*:

```

text = df['clean3']

factory = StemmerFactory()

stemming = factory.create_stemmer()

output = [(stemming.stem(token)) for token in text]

df['clean4'] = output

```

Gambar 3. 7 *Source code stemming*

- `text = df['clean3']`: mengambil atau memuat data teks yang telah dibersihkan dan disimpan dalam kolom 'clean3' DataFrame 'df' dan menyimpannya dalam variabel 'text'.
- `factory = StemmerFactory()`: Ini adalah bagian dari library Sastrawi yang digunakan untuk melakukan stemming pada teks dalam bahasa Indonesia. Kode ini membuat objek "factory" dari StemmerFactory.
- `stemming = factory.create_stemmer()`: Kode ini menggunakan objek "factory" untuk membuat objek "stemming" yang akan digunakan untuk melakukan proses stemming.
- `output = [(stemming.stem(token)) for token in text]`: Kode ini menggunakan objek "stemming" untuk menerapkan proses stemming pada setiap token (kata) dalam "text". Hasil stemming dari setiap kata disimpan dalam sebuah list yang disebut "output".
- `df['clean4'] = output`: Hasil stemming yang telah disimpan dalam "output" kemudian dimasukkan ke dalam kolom baru 'clean4' DataFrame 'df'. Ini berarti bahwa kolom 'clean4' akan berisi teks-teks dari kolom 'clean3' yang telah di-stem (dibawa ke bentuk dasarnya).

Tabel 3. 6 Hasil *stemming*

No	Clean 3	Clean 4
0	paketanku tgl stuck semarang	Paket tgl stuck semarang
1	harusnya hari sampe	Harus hari sampe
...
998	costumer nanyain kejelasan kurir malah diblok	Costumer nanya jelas kurir malah blok
999	coba jnt paketku skincare aman cuma lambat	Coba jnt paket skincare aman Cuma lambat

3.2.4 Checking

Setelah semua proses terlewati kemudian sebelum data benar-benar diproses data harus dipastikan dulu kevalidtannya dengan melihat jumlah label yang dihasilkan data tersebut harus seimbang dengan ketentuan maksimal kelas data adalah 60%:40% kemudian juga data harus dipastikan tidak ada yang eror dilihat apakah terdapat data duplikat dan data yang kosong pada *tweet* yang ada:

a. Check and Remove Blank Data

Proses ini merupakan pengecekan data *tweet* keseluruhan yang sudah melewati proses mulai dari *labelling* dan *cleaning*. Proses pengecekan ini dengan melihat apakah terdapat data kosong dari jumlah 1000 *tweet* yang dihasilkan dengan proses menggunakan *google colab* dengan bahasa pemrograman *python*. Jika terdapat data kosong atau *blank data* maka *machine* akan otomatis melakukan proses penghapusan pada kolom data yang kosong tersebut agar tidak mendapatkan eror atau *missing data* pada saat dilakukan pemrosesan dengan *source code* dan hasil berikut:

```
df.dropna(inplace=True)
df['tweet'].isna()
```

Gambar 3. 8 *Source code check and remove blank data*

Source code pada Gambar 3.8 diatas berisikan (`df.dropna(inplace=True)`) menghapus baris dengan nilai-nilai NaN dalam DataFrame 'df', mengubah DataFrame awal. Ini penting untuk membersihkan data yang tidak lengkap. Kemudian, kode kedua (`df['tweet'].isna()`) memeriksa apakah nilai dalam kolom 'tweet' adalah NaN, menghasilkan Series dengan nilai *True* jika NaN, *False* jika tidak. Hal ini berguna untuk melakukan validasi data dan memastikan integritasnya sebelum analisis lebih lanjut dengan hasil berikut:

```

0      False
1      False
2      False
3      False
4      False
...
995    False
996    False
997    False
998    False
999    False
Name: tweet, Length: 1000, dtype: bool

```

Gambar 3. 9 Hasil *check and remove blank data*

Terlihat hasil pemrosesan pada Gambar 3.9 diatas dari 1000 data yang ada, sudah dilakukan proses pengecekan data kosong atau *check blank data* tidak ditemukan data yang kosong pada setiap barisnya, dan jika menemui data yang kosong otomatis program akan menghapus baris yang terdeteksi.

b. Check and Remove Duplicate Data

Selanjutnya setelah semua kolom data dipastikan sudah terisi atau tidak ada data yang kosong, maka dilakukan pengecekan lagi untuk data yang ada lebih dari satu dengan isi teks yang sama atau *duplicate data*, dan jika terdapat data yang *duplicate* maka akan dilakukan penghapusan salah satu data *tweet* tersebut guna mendapatkan proses klasifikasi yang benar-benar valid dan proses juga dapat berjalan dengan cepat dengan *source code* dan hasil berikut:

```

df.drop_duplicates(inplace=True)
df.duplicated()

```

Gambar 3. 10 *Source code check and remove duplicate data*

Source code pada Gambar 3.10 diatas dengan penjelasan pada `df.drop_duplicates(inplace=True)`, kita hapus duplikat dalam DataFrame. Kemudian, `df.duplicated()` bantu identifikasi baris duplikat dengan Series *True* dan *False* dengan hasil berikut:

```

0      False
1      False
2      False
3      False
4      False
...
981    False
982    False
983    False
984    False
985    False
Length: 986, dtype: bool

```

Gambar 3. 11 Hasil *check and remove duplicate data*

Terlihat hasil pemrosesan pada Gambar 3.11 diatas hasil proses pengecekan dan penghapusan data awal setelah dilakukan proses *remove blank data* adalah sejumlah 1000 data kemudian setelah dilakukan proses *remove duplicate data* berkurang menjadi 986 dengan artian masih terdapat data kosong sejumlah 14 baris dan sudah dihapus secara otomatis oleh program. Dengan begitu maka data yang sudah didapatkan tidak lagi berisi data yang kosong (*blank data*) maupun data yang ganda (*duplicate data*) dalam artian data sudah benar-benar bersih dan siap untuk dilakukan pemrosesan tahap pengecekan data yang terakhir keseimbangan antar label (*imbalance data*).

c. Check Imbalance Data

Proses ini sangat harus diperhatikan dikarenakan jika tetap dilakukan proses klasifikasi pada label data yang tidak seimbang mesin yang membaca data akan mengklasifikasikan data secara berpihak pada salah satu kelas yang memiliki jumlah terbanyak saja. Fenomena ini dalam *machine learning* dikenal dengan istilah *imbalance dataset* yang mempunyai dampak buruk bagi model yang dibuat mengakibatkan model *overfitting* dengan beberapa penanganan yang dapat dilakukan yaitu *undersampling* atau *oversampling*.

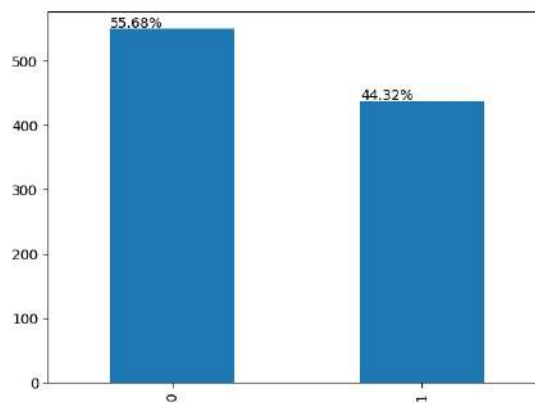
```

df['label'].value_counts()
s = pd.value_counts(df['label'])
ax = s.plot.bar()
n = len(df.index)
for p in ax.patches:
    ax.annotate(str(round(p.get_height() / n * 100, 2)) +
                '%', (p.get_x() * 1.005, p.get_height() * 1.005))

```

Gambar 3. 12 Source code check imbalance data

Source code pada Gambar 3.12 diatas menghasilkan grafik batang yang menunjukkan distribusi data dalam kolom 'label' dari DataFrame df, dengan persentase relatif disertakan sebagai anotasi di atas setiap batang hasil berikut:



Gambar 3. 13 Hasil check imbalance data

Terlihat hasil pada pemrosesan Gambar 3.13 diatas, hasil melakukan pengecekan otomatis berupa tampilan *interface* plot data dengan menggunakan pemrograman *python* dan perbandingan label positif digambarkan angka 1 yang mempunyai jumlah data sebanyak 437 data dengan persentase sebesar 44.32%, kemudian untuk label negatif digambarkan dengan angka 0 data berjumlah 549 dengan persentase sebesar 53.96%. Kesimpulan hasil dari pengecekan keseimbangan data didapatkan perbedaan antar label tidak terpaut jauh antara label positif dan label negatif, maka data tersebut masih tergolong data yang *balance* dan sudah memenuhi syarat untuk dilakukan pemrosesan ke tahap klasifikasi selanjutnya.

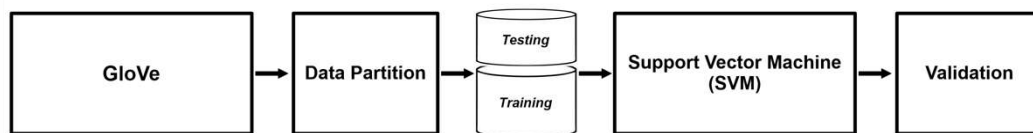
BAB IV

ANALISIS SENTIMEN BERBASIS GLOVE DAN SVM

Bagian bab ini membahas tentang penerapan dan hasil pengujian pertama teknik yaitu *Global Vectors for Word Representation* (GloVe) yang dipadukan dengan algoritma klasifikasi *Support Vector Machine* (SVM) dengan data yang sudah dipersiapkan pada tahap *data preparation* sebelumnya.

4.1 Deskripsi Penelitian

Pemrosesan menggunakan *google colab* serta *python* sebagai bahasa pemrograman yang digunakan, *library tensorflow* untuk pembuatan model *machine learning* dan dataset teks sebagai bahan uji dengan alur berikut:



Gambar 4. 1 Analisis Sentimen *GloVe* dan SVM

Terlihat pada Gambar 4.1 alur pengujian pada analisis sentimen berbasis *GloVe* dan SVM diatas akan dijelaskan secara umum pada langkah pertama adalah dengan *import library* yang dibutuhkan dan dataset yang sudah melalui proses *data preparation* kemudian membagi data tersebut menjadi dua bagian yaitu dengan perbandingan 80% data yang digunakan sebagai data *training* dan 20% data akan digunakan sebagai data *testing*, kemudian data yang sudah di bagi dua sebelumnya akan diubah kedalam nilai biner berbentuk vektor agar data dapat dibaca dan diproses oleh algoritma SVM pada *machine learning* proses inilah yang dinamakan *feature extraction* dengan keberhasilan dari proses tersebut akan diukur pada tahap akhir yaitu *validation* menggunakan *confusion matrix*.

4.2 Import Data dan Library

Langkah pertama adalah mempersiapkan data teks yang sudah melewati proses *data preparation* atau pembersihan dan validasi data, yang sudah benar-benar siap untuk dilakukan pemrosesan lebih lanjut yaitu klasifikasi teks pada tahap *feature extraction* menggunakan teknik *GloVe (Global Vector for Word Representation)* dengan memasukan beberapa *library* yang dibutuhkan dalam setiap rangkaian proses klasifikasi dengan *source code* berikut:

```
import csv
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import classification_report

data = pd.read_csv('/content/drive/MyDrive/Program Thesis/GLOVE
+ SVM/Data Bersih/Clean.csv')
data['tweet'] = data['tweet'].astype(str)
print(data.head())
```

Gambar 4. 2 Source code import library GloVe dan import data

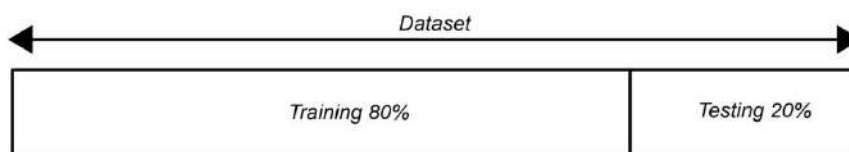
Source code pada Gambar 4.2 diatas adalah melakukan proses *import* beberapa *library* yang dibutuhkan yaitu pada baris pertama hingga baris ke tujuh, kemudian mengambil data bernama *clean.csv* dalam file komputer lokal yang diberi nama *data* dan selanjutnya file *data* tersebut dalam kolom *tweet* semua data dipastikan isinya bertipekan *string (str)*, menampilkan hasil lima data teratas berikut:

Tabel 4. 1 Hasil *import data*

No	label	tweet
0	0	paket tgl juni kirim cimahi semarang sampesamp...
1	1	siapsiap besok check out keranjang ya harbokir...
2	0	egk cok btw jnc tuh jincjancok sie ato
3	0	no kurir wilayah bogor selatan paket online am...
4	1	yaampun eonnie mmf mata ak buta

4.3 Data Partition

Langkah kedua adalah membagi data yang sudah bersih dan berlabel tersebut kedalam dua bagian yaitu data *training* dan data *testing*, pada pembagian data ini peneliti menggunakan aturan umum (*Rule of Thumb*) dalam persentase yaitu 80% digunakan sebagai data *training* dan 20% digunakan sebagai data *testing* dengan ilustrasi pada gambar dibawah:



Gambar 4. 3 Pembagian *training data* dan *testing data*

Training bagian dataset yang digunakan untuk melatih dan membuat prediksi atau menjalankan fungsi dari sebuah algoritma klasifikasi serta sebagai sumber data mesin yang kita latih untuk mencari korelasi pembelajaran pada pola data tersebut, sedangkan *testing* merupakan bagian dataset yang digunakan untuk menguji dan melihat keakuratan mesin yang sudah melakukan proses pembelajaran dan pengenalan pola yang dilakukan pada data *training* dengan *source code* berikut:

```
X = data['tweet']
y = data['label']

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2
,
random_state=42)
```

Gambar 4. 4 *Source code split data training dan testing*

Source code pada Gambar 4.4 merupakan proses pembagian data 'tweet' dan kemudian disimpan dalam variabel X, sementara untuk data 'label' disimpan kedalam variabel y. Dengan hasil akhir pembagian yang didapatkan yaitu sebanyak 788 data *training* dan 198 sebagai data *testing* dengan perbandingan 80%:20%.

4.4 *Global Vectors for Word Representation (GloVe)*

Merupakan sebuah metode dalam pengolahan bahasa alami manusia *Natural Language Processing* (NLP) dengan teknik untuk merepresentasikan kata-kata dalam bentuk vektor kedalam ruang multi dimensional dalam bentuk numerik agar dapat digunakan oleh algoritma komputer untuk pemahaman dan analisis teks yang biasa disebut dengan *feature extraction*. Model *GloVe* ini dikembangkan sekelompok peneliti dari *Stanford university* yang bernama Jeffrey Pennington, Richard Socher, dan Christopher Manning pada tahun 2014. Pada prinsipnya teknik ini bekerja dengan cara mengkombinasikan kedua hal terpenting dalam melakukan vektorisasi kata dalam matriks yaitu dengan melihat kata-kata disekitarnya secara makna (*Semantic*) dan melihat dari penting tidaknya kata tersebut berdasarkan seberapa sering muncul (*Count*).

Langkah pertama dalam penerapan teknik *GloVe* ini adalah dengan membangun *co-occurrence matrix*. *Co-occurrence matrix* bekerja dengan cara merekam seberapa sering pasangan kata muncul bersama dalam konteks yang sama dalam korpus atau kamus teks yang dianalisis. Matriks ini membantu menggambarkan hubungan statistik antara kata-kata dalam teks dan akan digunakan sebagai dasar untuk menghasilkan representasi vektor kata kedalam ruang vektor dengan menggunakan persamaan rumus, kemudian menghitung nilai persamaan rumus tersebut untuk setiap pasangan kata yang mencerminkan kekuatan hubungan antar kata berdasarkan frekuensi kemunculan. Dengan menggunakan bobot-bobot ini, *GloVe* melakukan iterasi untuk memperbaiki representasi vektor agar menghasilkan kata yang lebih informatif dan berarti dalam ruang vektor yang telah dibangun dengan alur rumus berikut:

Dokumen yang terdiri dari N kata, dan membuat *co-occurrence* matrix N

- Kata ke- i adalah w_i untuk $i = 1, 2, \dots, N$.
- Kata ke- j adalah w_j untuk $j = 1, 2, \dots, N$.

Maka untuk mengisi *co-occurrence* matrix X dimana X_{ij} adalah elemen pada baris ke- i dan kolom ke- j dapat dihitung sebagai berikut:

X_{ij} = Jumlah kemunculan Bersama (w_i, w_j)

Dalam Bahasa matematis:

$$X_{ij} = \sum_{k=1}^K \delta(w_i, w_{wi,k}) \quad (4.1)$$

Dimana $\delta(w_i, w_{wi,k})$ adalah fungsi delta yang bernilai 1 jika w_i dan w_j muncul bersama dalam konteks yang sama dalam salah satu dokumen dari korpus, dan 0 jika tidak, dengan hasil penerapan rumus tersebut pada *source code* berikut:

```
unique_words = list(set(tokens_list))
co_occurrence_matrix = np.zeros((len(unique_words),
len(unique_words)), dtype=int)

for i, word1 in enumerate(unique_words):
    for j, word2 in enumerate(unique_words):
        if i != j:
            co_occurrence_matrix[i, j] = tokens_list.count(word1 + " "
+ word2)
```

Gambar 4. 5 Source code co-occurrence matrix

Source code pada Gambar 4.5 digunakan untuk membuat matriks *co-occurrence* dari daftar sekumpulan kata unik dalam teks. Prosesnya melibatkan pembentukan daftar kata unik, inialisasi matriks nol, lalu penghitungan *co-occurrence* antara kata unik dalam jarak satu kata sebelum dan setelahnya, dengan menerapkan dan mengikuti alur pada persamaan yang sudah dipaparkan atau dijelaskan sebelumnya. Dengan hasil dan tujuannya untuk mencari dan menganalisis hubungan antar kata dalam teks dokumen besar yang telah diproses.

Berikut adalah hasil dari pemrosesan dari penerapan *source code* untuk *co-occurrence matrix* yang disajikan pada Tabel 4.2 dengan mengambil 2 data teratas dari 986 data yang telah melewati proses *reprocessing* atau pembersihan data hingga siap diproses yaitu “Paket tgl stuck semarang” Dan "Harus hari sampe" berikut hasil tabelnya:

Tabel 4. 2 Hasil *co-occurrence matrix*

	paket	tlg	stuck	semarang	harus	hari	sampe
paket	0	1	1	1	0	0	0
tlg	1	0	0	0	0	0	0
stuck	1	0	0	1	0	0	0
semarang	1	0	1	0	0	0	0
harus	0	0	0	0	0	1	1
hari	0	0	0	0	1	0	1
sampe	0	0	0	0	1	1	0

Dalam Tabel 4.2 baris dan kolom mewakili kata-kata yang ada dalam *vocabulary*, dan nilai dalam sel tabel mengindikasikan berapa kali pasangan kata tersebut muncul bersama dalam konteks yang sama. Misalnya, pasangan "paket" dan "tgl" muncul bersama dalam konteks dalam 1 kalimat, sehingga nilai di baris "paket" dan kolom "tgl" adalah 1. Dimana akan mengembangkan model matematis untuk menghasilkan representasi vektor kata yang lebih kaya makna dan menggambarkan hubungan antara kata-kata dalam ruang vektor.

Kemudian proses selanjutnya membaca dan memuat vektor kata dari file teks yang berisi representasi kata *GloVe* berbentuk nilai-nilai angka berupa vektor yang dilanjutkan kedalam kamus *python* dengan tujuan untuk memungkinkan mesin memahami hubungan antar kata. Implementasi *source code* yang diterapkan dapat membantu menyederhanakan proses tersebut, memastikan integrasi yang efisien dalam pemodelan analisis sentimen berbasis teks dalam membangun dasar pemahaman kata untuk meningkatkan kinerja pada *machine*.

```

glove_embeddings = {}
with open('/content/drive/MyDrive/Program Thesis/GLOVE +
SVM/Data Bersih/glove.42B.300d.txt', encoding='utf-8') as f: #
path ke file GloVe
    for line in f:
        values = line.split()
        word = values[0]
        vector = np.asarray(values[1:], dtype='float32')
        glove_embeddings[word] = vector

word_vector = glove_embeddings.get("cepat")
if word_vector is not None:
    print(word_vector)
else:
    print("Kata 'cepat' tidak ditemukan dalam kamus.")

```

Gambar 4. 6 Source code proses vektorisasi kata

Jadi, secara keseluruhan *source code* pada Gambar 4.6 diatas adalah membaca file *GloVe* serta memproses setiap baris untuk memisahkan kata dari vektor dan kemudian menyimpannya dalam kamus *glove_embeddings*. Setelah *source code* ini selesai dijalankan, maka akan memiliki akses ke vektor *GloVe* untuk kata-kata tertentu dalam bentuk kamus. Kemudian hasil dari vektorisasi dari setiap kata pada seluruh dokumen akan dapat terlihat, sebagai contoh kita akan melihat vektor untuk kata “cepat” seperti pada *source code* diatas dengan hasil berikut:

```

[ 2.8280e-01 -2.7152e-01  2.2248e-01 -1.8093e-01  3.7818e-01 -3.1264e-01
 3.9155e-01  3.7203e-01  1.9905e-01  5.0884e-02  6.7022e-01  3.1414e-01
 4.7979e-01  4.3670e-01 -1.1677e-01  4.0018e-01  3.4184e-01  5.0982e-01
 2.3216e-01  2.8953e-01  1.0505e-01  8.3718e-01 -4.4335e-01 -7.1572e-01
-3.5597e-02  4.3626e-01 -3.5043e-01  5.2458e-01 -1.7076e-01 -8.0248e-02
-1.4023e-01 -1.9521e-01  9.2540e-01 -7.1706e-02  3.3433e-01  2.2378e-03
 1.3349e-01  1.5801e-01  2.7931e-01  8.9495e-02  5.4694e-02 -7.0885e-01
 9.7183e-01 -4.4399e-01 -2.7988e-01 -6.4673e-01 -5.4296e-01  3.0757e-01
-1.9089e-02  6.0583e-02 -7.8873e-01 -1.3792e-01  2.2902e-01 -1.4480e-01
-6.0583e-01  7.5678e-01 -2.8515e-01 -3.5930e-02 -1.0188e-02 -5.3247e-01
 3.0726e-01  1.9859e-01 -1.4728e-01 -8.2912e-01 -4.8605e-01 -1.2444e-01
 2.8522e-01  6.9131e-01 -9.3932e-02  1.4201e-01  4.9556e-02 -3.2201e-01
 2.9920e-01  7.3191e-01  3.0296e-01  1.8200e-01  7.8775e-01 -4.0277e-01
-7.2793e-01 -1.0257e+00 -4.7461e-01  9.6553e-02  5.8088e-01  1.8981e-01
-7.3810e-01  5.6169e-02  1.3762e-01  9.2372e-02  2.6115e-01  2.0583e-01
 7.0095e-02  3.5248e-01 -6.4255e-01  2.4332e-01  7.5319e-02  2.0649e-01
 5.4745e-01  4.1856e-01 -3.0273e-01  4.7594e-01 -7.1840e-01 -1.4995e-01
-3.0557e-01  1.7123e-01  2.4196e-01 -5.3148e-01  3.0974e-01  7.5910e-01
 4.2978e-02  4.2550e-01  1.9746e-01 -5.2569e-01  5.8747e-01  2.3919e-01
-4.4764e-01  2.7972e-01  3.8390e-02 -4.9496e-01  6.0945e-02 -7.8634e-02
 1.0441e-01  2.5007e-01  4.8317e-01  4.9072e-01 -3.2905e-01  4.0142e-01
-6.8528e-01 -2.8580e-01 -1.6558e-01 -5.5026e-01 -1.7321e-01  6.6402e-01

```

5.3563e-02	9.0485e-03	-9.6030e-01	-4.7625e-01	3.3723e-01	-2.9530e-02
1.0917e-02	3.4473e-01	6.2329e-01	-2.7767e-01	-2.2958e-01	-2.6191e-01
5.0815e-02	2.9424e-01	-2.6352e-02	-3.6346e-01	-1.7406e-02	-1.1513e-01
2.9816e-01	-2.1238e-01	3.2303e-01	5.1967e-01	-4.5752e-01	3.7455e-01
4.5436e-01	-5.0839e-03	8.4000e-01	-7.7313e-02	8.8564e-01	-3.9812e-01
-4.8523e-01	2.8163e-01	1.0763e-02	-4.0237e-01	1.3559e-01	-3.0111e-01
3.6577e-01	2.4911e-01	2.5212e-01	5.3581e-01	1.6037e-01	6.1491e-03
-3.6072e-01	-6.8317e-01	-4.4592e-01	7.0361e-01	-1.3422e-01	-2.8943e-01
4.7948e-02	2.2563e-01	-1.1568e-01	2.9458e-01	-3.3287e-02	8.5112e-03
-3.8304e-02	6.0385e-01	4.3842e-01	5.1238e-01	-1.0745e-01	-4.6659e-01
4.4545e-01	3.0590e-01	6.9391e-02	4.1035e-01	7.6794e-02	-4.3645e-01
-2.5872e-01	-7.1711e-02	-3.8567e-01	-5.7782e-01	4.8681e-01	5.3069e-01
-4.1602e-01	5.8898e-01	4.5243e-01	3.4697e-01	-3.5794e-01	-4.0762e-01
3.5440e-01	2.8984e-01	2.5330e-01	-1.8601e-01	5.7050e-01	-5.1691e-01
1.6503e-01	-1.5174e-01	1.4987e-01	-9.9405e-01	-3.2538e-01	2.9497e-01
-1.3193e-01	-3.4766e-01	3.8362e-01	-1.9747e-01	-1.1643e-01	3.8846e-01
-3.3770e-01	3.2565e-02	5.8245e-01	2.4617e-01	3.9397e-01	2.5406e-01
6.2994e-01	7.5558e-02	-1.1547e-02	-5.8511e-01	3.0191e-01	-3.4368e-02
4.6986e-01	2.6761e-01	-1.9474e-01	4.5717e-04	4.5111e-01	7.3816e-02
-9.6759e-02	-2.0098e-01	-1.1678e-01	3.0287e-01	1.3923e-01	-1.1224e+00
2.4196e-01	-8.7351e-01	1.4972e-01	2.8015e-01	-7.5237e-02	1.7709e-01
-5.3001e-01	-4.7126e-01	-2.2813e-01	5.0409e-01	-4.6830e-01	-9.0496e-02
1.2429e-01	1.2788e-01	6.3138e-02	7.2587e-01	1.9227e-01	2.6171e-01
-6.2209e-01	-7.1862e-01	-2.1574e-01	3.6895e-02	-1.5902e-01	1.1389e+00
3.6080e-01	3.0054e-01	-3.4154e-01	-1.6561e-01	-1.9290e-01	-8.0080e-02
-4.2154e-02	2.1481e-01	-3.8524e-01	7.3883e-01	-5.0500e-01	-3.3618e-01
-6.7935e-02	3.8533e-01	3.9389e-01	3.1992e-02	-4.0497e-01	6.0265e-02
4.3842e-01	-4.0926e-02	-8.5671e-01	-9.2808e-01	1.1753e-01	-5.8733e-01]

Gambar 4. 7 Hasil vektorisasi kata “cepat”

Langkah ketiga atau langkah terakhir yang dilakukan setelah memuat vektor *GloVe* ke dalam kamus, serta *source code* ini digunakan untuk menghasilkan vektor representasi teks untuk kumpulan data pelatihan (X_{train}) dan pengujian (X_{test}).

Berikut *source code* beserta penjelasan lengkapnya:

```
def get_text_embeddings(text):
    text_tokens = text.split()
    embeddings = [glove_embeddings[word] for word in text_tokens
if word in glove_embeddings]
    if embeddings:
        return np.mean(embeddings, axis=0)
    else:
        return np.zeros(300)

X_train_glove = np.array([get_text_embeddings(text) for text in
X_train])
X_test_glove = np.array([get_text_embeddings(text) for text in
X_test])
```

Gambar 4. 8 *Source code* representasi vektor berdasarkan setiap dokumen

Dengan *source code* pada Gambar 4.8, telah menghasilkan vektor representasi *GloVe* untuk seluruh teks dalam kumpulan data pelatihan dan pengujian. Ini memungkinkan untuk menggunakan vektor *GloVe* sebagai fitur dalam model pemrosesan bahasa alami atau pembelajaran mesin untuk berbagai tugas klasifikasi teks, berikut adalah pengecekan hasil akhir untuk vektor yang didapat:

```
print("Hasil akhir vektor GloVe untuk data pelatihan
(X_train_combined):")
print(X_train_combined)
```

Gambar 4. 9 *source code* menampilkan vektor *GloVe*

Dari *source code* pada Gambar 4.9 digunakan untuk menampilkan hasil akhir dari pemrosesan *feature extraction* untuk teknik *GloVe* dengan data 3 teratas dan 3 terahir serta dibuktikan pada hasil berikut:

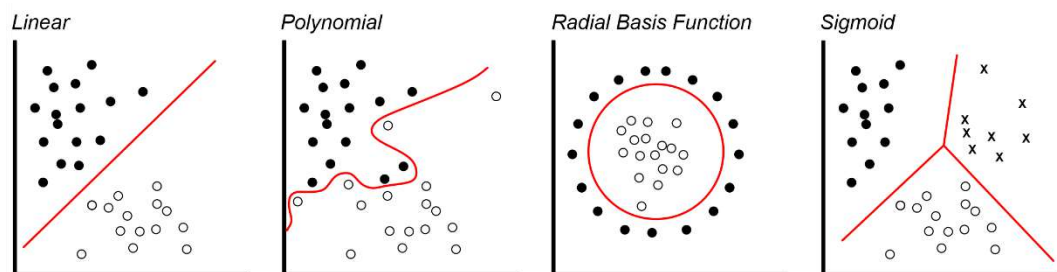
```
[[ 0.17783278 -0.27532348 0.09177171 ... -0.30226731 -0.02045235 -0.05867546]
 [ 0.03680317 -0.16946749 0.22413917 ... -0.07621475 0.03201191 -0.07463342]
 [-0.05034725 -0.00934825 0.17824975 ... 0.1653645 0.04434 0.036825 ]
 ...
 [-0.02075275 -0.13479075 -0.00144575 ... -0.38608149 0.28939474 -0.10437 ]
 [ 0.02425373 -0.08131443 0.21451676 ... -0.04096412 0.03655874 -0.03699863]
 [-0.0823625 0.00116999 -0.032832 ... -0.37050003 0.38437 0.19776215]]
```

Gambar 4. 10 Hasil *feature extraction GloVe*

Feature extraction pada Gambar 4.10 ini menghasilkan matriks yang mewakili struktur dan makna kata dalam dokumen. Dengan matriks ini, dapat melakukan analisis semantik, klustering, atau perbandingan dokumen dengan analisis sentimen menggunakan pola *machine learning* berdasarkan representasi vektor kata-kata yang dihasilkan oleh model *GloVe* yang telah didapatkan, kemudian langkah selanjutnya adalah data berupa angka berbentuk vektor tersebut dilakukan pemrosesan atau pengklasifikasian menggunakan model algoritma yang telah dipilih yaitu *Support Vector Machine* (SVM) untuk mengklasifikasikan data berdasarkan sentiment negatif dan sentimen positif dari nilai vektor yang telah didapatkan oleh pemrosesan *feature extratrion GloVe*.

4.5 Support Vector Machine (SVM)

Setelah data melewati proses *splitting* dan proses *feature extraction*, saat ini data menjadi bentuk numerik berupa vektor lalu pada tahap inilah dilakukan proses modeling klasifikasi dengan menerapkan metode *Support Vector Machine* (SVM). Algoritma pembelajaran ini pertama kali dikembangkan oleh Boser, Guyon dan Vapnik pada tahun 1992 yang merupakan sistem pembelajaran dengan menggunakan hipotesis berupa fungsi-fungsi linier didalam sebuah fitur yang memiliki dimensi tinggi dan dilatih menggunakan teori optimasi.



Gambar 4. 11 Algoritma *Support Vector Machine* (SVM)

Pada dasarnya konsep dan cara kerja dari metode *Support Vector Machine* (SVM) adalah berusaha mencari dan menemukan fungsi garis pemisah (*hyperplane*) yang terbaik diantara beberapa fungsi. Seperti pada Gambar 4.11 terdapat 4 fungsi atau karnel dengan tipe data dan pola masing-masing dan masing-masing memiliki jenis garis pemisah yang berbeda-beda pada setiap fungsinya. Metode *Support Vector Machine* (SVM) mencari dan memilih dari salah-satu garis/*hyperplane* yang paling baik dan pemilihan tersebut sangat bergantung pada jenis data dengan tujuan dapat memisahkan antar kelas yang berbeda dengan sempurna. Formulasi algoritma ini dapat dibedakan menjadi beberapa fungsi atau karnel yang digunakan yaitu *linear*, *polynomial*, *radial basis function* dan *sigmoid* dengan cara mencari dan mengoptimalkan pemrosesan yaitu proses *tunning*.

Langkah awal adalah *tuning SVM (Support Vector Machine)* dalam *machine learning* adalah proses untuk mengoptimalkan parameter-parameter model SVM agar memberikan kinerja yang terbaik untuk tugas yang sedang dihadapi. Berikut adalah langkah-langkah dalam proses tuning SVM dalam penerapan *source code*:

```
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf', 'linear', 'poly', 'sigmoid']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

grid.fit(X_train_combined, y_train)
```

Gambar 4. 12 *Source code tuning SVM*

- a. Pemilihan karnel, memilih jenis kernel function yang akan digunakan untuk menentukan cara SVM akan memetakan data ke ruang fitur yang lebih tinggi serta yang digunakan *linear*, *polynomial*, *Radial Basis Function (RBF)* dan *sigmoid*. Pemilihan kernel sangat berpengaruh besar pada hasil model didapat.
- b. Penentuan parameter C, dalam mengatur *trade-off* antara margin yang lebih besar dan kesalahan klasifikasi yang lebih kecil. Nilai C yang lebih besar akan memberikan margin yang lebih kecil tetapi kesalahan klasifikasi yang lebih kecil, sementara nilai C yang lebih kecil akan memberikan margin yang lebih besar tetapi kesalahan klasifikasi yang lebih besar. Ini adalah langkah penting dalam tuning SVM, dengan metode *GridSearch* untuk menemukan nilai terbai
- c. Penentuan parameter gamma, Parameter ini mengontrol fleksibilitas model terhadap data pelatihan. Dengan berpatokan pada hasil dengan gambaran nilai gamma yang lebih besar dapat menghasilkan model yang lebih rumit, sementara nilai gamma yang lebih kecil dapat menghasilkan model yang lebih sederhana. Sama seperti parameter C, serta dapat mencoba berbagai nilai gamma untuk menemukan hasil yang optimal.

Berikut merupakan hasil tuning yang dilakukan *machine* terhadap data yang telah diproses dengan berbagai parameter yang telah di masukan sebagai pilihan untuk mencari model SVM yang paling optimal:

```

Fitting 5 folds for each of 100 candidates, totalling 500 fits
[CV 1/5] END .....C=1, gamma=1, kernel=rbf;; score=0.544 total time= 0.1s
[CV 2/5] END .....C=1, gamma=1, kernel=rbf;; score=0.570 total time= 0.1s
[CV 3/5] END .....C=1, gamma=1, kernel=rbf;; score=0.570 total time= 0.1s
[CV 4/5] END .....C=1, gamma=1, kernel=rbf;; score=0.592 total time= 0.1s
[CV 5/5] END .....C=1, gamma=1, kernel=rbf;; score=0.586 total time= 0.1s
[CV 1/5] END .....C=1, gamma=1, kernel=linear;; score=0.797 total time= 0.0s
[CV 2/5] END .....C=1, gamma=1, kernel=linear;; score=0.772 total time= 0.0s
[CV 3/5] END .....C=1, gamma=1, kernel=linear;; score=0.797 total time= 0.0s
[CV 4/5] END .....C=1, gamma=1, kernel=linear;; score=0.803 total time= 0.0s
[CV 5/5] END .....C=1, gamma=1, kernel=linear;; score=0.790 total time= 0.0s
[CV 1/5] END .....C=1, gamma=1, kernel=poly;; score=0.759 total time= 0.1s
[CV 2/5] END .....C=1, gamma=1, kernel=poly;; score=0.816 total time= 0.1s
[CV 3/5] END .....C=1, gamma=1, kernel=poly;; score=0.791 total time= 0.1s
[CV 4/5] END .....C=1, gamma=1, kernel=poly;; score=0.790 total time= 0.1s
[CV 5/5] END .....C=1, gamma=1, kernel=poly;; score=0.854 total time= 0.1s
[CV 1/5] END .....C=1, gamma=1, kernel=sigmoid;; score=0.361 total time= 0.1s
[CV 2/5] END .....C=1, gamma=1, kernel=sigmoid;; score=0.443 total time= 0.1s
[CV 3/5] END .....C=1, gamma=1, kernel=sigmoid;; score=0.449 total time= 0.1s
[CV 4/5] END .....C=1, gamma=1, kernel=sigmoid;; score=0.395 total time= 0.1s
[CV 5/5] END .....C=1, gamma=1, kernel=sigmoid;; score=0.408 total time= 0.1s
[CV 1/5] END .....C=1, gamma=0.1, kernel=rbf;; score=0.880 total time= 0.1s
[CV 2/5] END .....C=1, gamma=0.1, kernel=rbf;; score=0.804 total time= 0.1s
[CV 3/5] END .....C=1, gamma=0.1, kernel=rbf;; score=0.741 total time= 0.1s
[CV 4/5] END .....C=1, gamma=0.1, kernel=rbf;; score=0.822 total time= 0.1s
[CV 5/5] END .....C=1, gamma=0.1, kernel=rbf;; score=0.841 total time= 0.1s
[CV 1/5] END .....C=1, gamma=0.1, kernel=linear;; score=0.797 total time= 0.0s
[CV 2/5] END .....C=1, gamma=0.1, kernel=linear;; score=0.772 total time= 0.0s
[CV 3/5] END .....C=1, gamma=0.1, kernel=linear;; score=0.797 total time= 0.0s
[CV 4/5] END .....C=1, gamma=0.1, kernel=linear;; score=0.803 total time= 0.0s
[CV 5/5] END .....C=1, gamma=0.1, kernel=linear;; score=0.790 total time= 0.0s
[CV 1/5] END .....C=1, gamma=0.1, kernel=poly;; score=0.854 total time= 0.1s
[CV 2/5] END .....C=1, gamma=0.1, kernel=poly;; score=0.778 total time= 0.1s
[CV 3/5] END .....C=1, gamma=0.1, kernel=poly;; score=0.753 total time= 0.1s
[CV 4/5] END .....C=1, gamma=0.1, kernel=poly;; score=0.828 total time= 0.0s
[CV 5/5] END .....C=1, gamma=0.1, kernel=poly;; score=0.822 total time= 0.0s
[CV 1/5] END ....C=1, gamma=0.1, kernel=sigmoid;; score=0.778 total time= 0.1s
[CV 2/5] END ....C=1, gamma=0.1, kernel=sigmoid;; score=0.791 total time= 0.1s
[CV 3/5] END ....C=1, gamma=0.1, kernel=sigmoid;; score=0.791 total time= 0.1s
[CV 4/5] END ....C=1, gamma=0.1, kernel=sigmoid;; score=0.764 total time= 0.1s
[CV 5/5] END ....C=1, gamma=0.1, kernel=sigmoid;; score=0.796 total time= 0.1s
[CV 1/5] END .....C=1, gamma=0.01, kernel=rbf;; score=0.816 total time= 0.1s
[CV 2/5] END .....C=1, gamma=0.01, kernel=rbf;; score=0.785 total time= 0.1s
[CV 3/5] END .....C=1, gamma=0.01, kernel=rbf;; score=0.703 total time= 0.1s
[CV 4/5] END .....C=1, gamma=0.01, kernel=rbf;; score=0.790 total time= 0.1s
[CV 5/5] END .....C=1, gamma=0.01, kernel=rbf;; score=0.783 total time= 0.1s
[CV 1/5] END ....C=1, gamma=0.01, kernel=linear;; score=0.797 total time= 0.0s
[CV 2/5] END ....C=1, gamma=0.01, kernel=linear;; score=0.772 total time= 0.0s
[CV 3/5] END ....C=1, gamma=0.01, kernel=linear;; score=0.797 total time= 0.0s
[CV 4/5] END ....C=1, gamma=0.01, kernel=linear;; score=0.803 total time= 0.0s
[CV 5/5] END ....C=1, gamma=0.01, kernel=linear;; score=0.790 total time= 0.0s
[CV 1/5] END .....C=1, gamma=0.01, kernel=poly;; score=0.557 total time= 0.1s
[CV 2/5] END .....C=1, gamma=0.01, kernel=poly;; score=0.557 total time= 0.1s
[CV 3/5] END .....C=1, gamma=0.01, kernel=poly;; score=0.557 total time= 0.1s
[CV 4/5] END .....C=1, gamma=0.01, kernel=poly;; score=0.561 total time= 0.1s
[CV 5/5] END .....C=1, gamma=0.01, kernel=poly;; score=0.561 total time= 0.1s

```



```

[CV 5/5] END .....C=10, gamma=1, kernel=poly;; score=0.854 total time= 0.1s
[CV 1/5] END .....C=10, gamma=1, kernel=sigmoid;; score=0.348 total time= 0.1s
[CV 2/5] END .....C=10, gamma=1, kernel=sigmoid;; score=0.405 total time= 0.1s
[CV 3/5] END .....C=10, gamma=1, kernel=sigmoid;; score=0.443 total time= 0.1s
[CV 4/5] END .....C=10, gamma=1, kernel=sigmoid;; score=0.357 total time= 0.1s
[CV 5/5] END .....C=10, gamma=1, kernel=sigmoid;; score=0.420 total time= 0.1s
[CV 1/5] END .....C=10, gamma=0.1, kernel=rbf;; score=0.810 total time= 0.1s
[CV 2/5] END .....C=10, gamma=0.1, kernel=rbf;; score=0.842 total time= 0.1s
[CV 3/5] END .....C=10, gamma=0.1, kernel=rbf;; score=0.778 total time= 0.1s
[CV 4/5] END .....C=10, gamma=0.1, kernel=rbf;; score=0.815 total time= 0.1s
[CV 5/5] END .....C=10, gamma=0.1, kernel=rbf;; score=0.841 total time= 0.1s
[CV 1/5] END ....C=10, gamma=0.1, kernel=linear;; score=0.785 total time= 0.1s
[CV 2/5] END ...C=10, gamma=0.1, kernel=linear;; score=0.753 total time= 0.1s
[CV 3/5] END ...C=10, gamma=0.1, kernel=linear;; score=0.804 total time= 0.1s
[CV 4/5] END ...C=10, gamma=0.1, kernel=linear;; score=0.771 total time= 0.1s
[CV 5/5] END ...C=10, gamma=0.1, kernel=linear;; score=0.803 total time= 0.1s
[CV 1/5] END .....C=10, gamma=0.1, kernel=poly;; score=0.861 total time= 0.1s
[CV 2/5] END .....C=10, gamma=0.1, kernel=poly;; score=0.829 total time= 0.1s
[CV 3/5] END .....C=10, gamma=0.1, kernel=poly;; score=0.804 total time= 0.1s
[CV 4/5] END .....C=10, gamma=0.1, kernel=poly;; score=0.828 total time= 0.1s
[CV 5/5] END .....C=10, gamma=0.1, kernel=poly;; score=0.854 total time= 0.1s

```

Gambar 4. 13 Hasil *tunning* SVM

Pada penerapan model SVM yang digunakan dan telah dilakukan *tunning* dengan hasil pemrosesan *machine* seperti pada Gambar 4.13 telah didapatkan hasil yang paling optimal pada nilai $C=10$, $\gamma=0.1$ dan $\text{kernel}=\text{polynomial}$. Dengan cara mencari keluaran hasil pencarian terbaik menggunakan *grid.best_params_*.

```
print(grid.best_params_)
```

Gambar 4. 14 Source code show best *tunning*

Dengan hasil output berikut:

```
{'C': 10, 'gamma': 0.1, 'kernel': 'poly'}
```

Gambar 4. 15 Output best *tunning*

Setelah memperoleh hasil *tuning* untuk model yang telah ditetapkan yaitu *Support Vector Machine* (SVM), langkah berikutnya adalah menampilkan hasil nilai visual berupa grafik berbentuk tabel dengan melibatkan pemahaman visual yang mendalam terhadap model yang telah disusun. Ini mencakup analisis nilai C , evaluasi hasil nilai γ , dan pemahaman terperinci terkait jenis kernel yang digunakan pada parameter ini dapat memahami dampaknya terhadap kinerja keseluruhan model *Support Vector Machine* (SVM).

Pada dasarnya, *Support Vector Machine* SVM merupakan suatu *linear classifier*. Namun, SVM dapat dikembangkan menjadi *nonlinear classifier* dengan konsep kernel *trick* pada ruang berdimensi lebih tinggi dapat menangani *nonlinear classifier*. terdapat fungsi kernel *trick* pada SVM yang sudah didapatkan pada proses *tunning* sebelumnya adalah karnel *polynomial* dengan persamaan berikut:

$$K(x_i, x_j) = ((x_i, x_j) + 1)^d \quad (4.2)$$

Dengan:

K = Fungsi karnel

x_i, x_j = Fektor dari dataset

d = Pangkat *polynomial*

Kemudian untuk langkah-langkah persamaan secara umum menggunakan metode *Support Vector Machine* (SVM) sebagai berikut:

1. Merepresentasikan kata-kata dalam bentuk vektor numerik berdasarkan distribusi kata-kata cuitan twitter dalam konteks korpus teks *GloVe*;
2. Menginisialisasi berbagai parameter yang diperlukan dalam perhitungan manual dengan SVM seperti a_i , γ , C , s , λ , dan i_{max} ;

Keterangan:

a_i = Alpha atau *lagrange multiplier* untuk mencari *support vector*

γ = *Learning rate* untuk mengontrol kecepatan

C = *Cost* untuk meminimalkan nilai *error* saat proses *training*

s = Epsilon untuk ukuran tingkat *error* klasifikasi

λ = Turunan batas teoritis

i_{max} = Iterasi maksimum

3. Melakukan perhitungan kernel *polynomial* terlebih dahulu;
4. Melakukan perhitungan terhadap nilai y atau label dari kelas yang telah ditentukan yaitu kelas positif (1) dan kelas negatif(0);
5. Menghitung matriks *Hessian* dengan menggunakan persamaan berikut:

$$D_{ij} = y_i y_j (K(x_i, x_j) + \lambda^2) \quad (4.3)$$

Dengan nilai i dan $j = 1, 2, 3, \dots, n$.

Keterangan:

D = matriks *Hessian*

y_i = kelas data ke- i

y_j = kelas data ke- j

$K(x_i, x_j)$ = fungsi kernel yang digunakan

6. Menghitung nilai *error rate* dengan tujuan untuk mencerminkan seberapa sering model membuat kesalahan dalam memprediksi kelas atau nilai target.

$$E_i = \sum_{j=1}^n a_j D_{ij} \quad (4.4)$$

Keterangan:

E_i = nilai *error rate* data ke- i

7. Menghitung nilai delta alpha untuk melihat perubahan fungsi yang disebut dengan *Lagrange Multiplier*. Jika data *training* telah mencapai nilai konvergen ($\max(|\delta a_i|) < \mathcal{E}$) dan ketika maksimum iterasi mencapai nilai yang ditentukan, maka iterasi akan berhenti;

$$\delta a_i = \min(\max[\gamma(1 - E_i), a_i], C - a_i) \quad (4.5)$$

Keterangan:

δa_i = nilai delta alfa data ke- i

8. Menghitung nilai a_i baru, koefisien yang baru dihitung sebagai bagian dari vektor bobot (w) yang menggambarkan *hyperplane* pemisah agar dapat digunakan pada iterasi selanjutnya;

$$a_i = a_i + \delta a_i \quad (4.6)$$

9. Menghitung nilai $w \cdot x^+$ dan $w \cdot x^-$ untuk mendapatkan nilai bias b atau parameter yang bertanggung jawab untuk menentukan posisi *hyperplane*;

$$w \cdot x^+ = a_i y_i K(x_i, x^+) \quad (4.7)$$

$$w \cdot x^- = a_i y_i K(x_i, x^-) \quad (4.8)$$

$$b = -\frac{1}{2} (w \cdot x^+ + w \cdot x^-) \quad (4.9)$$

Keterangan:

$w \cdot x^+$ = nilai kernel data x dengan data x kelas positif

$w \cdot x^-$ = nilai kernel data x dengan data x kelas negatif

b = nilai bias

10. Setelah mendapatkan nilai bias, lalu menghitung kernel dan nilai bobot dari data *testing* agar dapat dimasukkan ke dalam fungsi klasifikasi dari kedua data *testing* untuk menentukan kelas data uji;

$$f(x) = \text{sign} \sum_{i=0}^n (a_i y_i K(x, x_i) + b) \quad (4.10)$$

Keterangan:

w = parameter *hyperplane* yang dicari garis tegak lurus antara garis *hyperplane* dan titik *support vector*

x = titik data masukan *Support Vector Machine*

a_i = nilai bias

$K(x, x_i)$ = fungsi kernel

Berikut adalah *source code* visualisasi tabel hasil dari tuning yang telah dilakukan model SVM dengan mencari hasil nilai yang paling tinggi:

```
# Definisikan parameter yang ingin Anda jelajahi
C_range = [0.1, 1, 10, 100, 1000]
gamma_range = [0.0001, 0.001, 0.01, 0.1, 1]
kernels = ['rbf', 'linear', 'poly', 'sigmoid']

# Inisialisasi array untuk menyimpan metrik performa
performa = np.zeros((len(C_range), len(gamma_range),
len(kernels)))

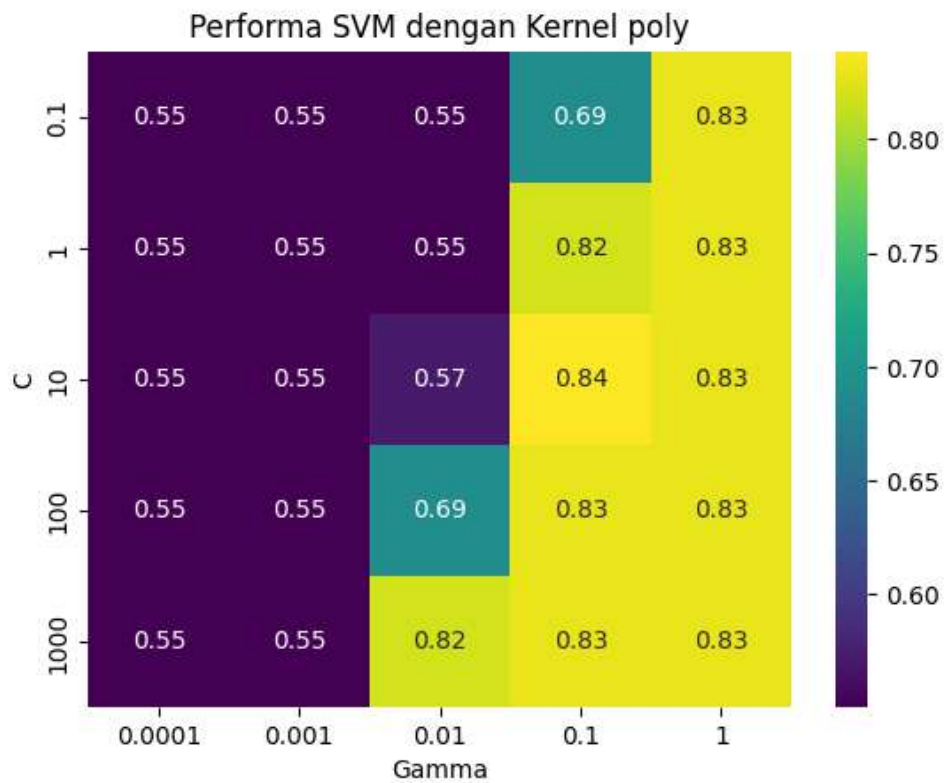
# Loop untuk mengeksekusi GridSearchCV pada setiap kombinasi
parameter
for i, C in enumerate(C_range):
    for j, gamma in enumerate(gamma_range):
        for k, kernel in enumerate(kernels):
            svc = SVC(C=C, gamma=gamma, kernel=kernel)
            svc.fit(X_train_combined, y_train)
            performa[i, j, k] = svc.score(X_test_combined, y_test)

# Buat heatmap untuk setiap kernel
for k, kernel in enumerate(kernels):
    plt.figure()
    sns.heatmap(performa[:, :, k], annot=True,
xticklabels=gamma_range, yticklabels=C_range, cmap='viridis')
    plt.title(f'Performa SVM dengan Kernel {kernel}')
    plt.xlabel('Gamma')
    plt.ylabel('C')
    plt.show()
```

Gambar 4. 16 *Source code visualisasi params*

Source code pada Gambar 4.16 adalah melakukan pemindaian grid (*grid search*) pada model *Support Vector Machine* (SVM) dengan berbagai kombinasi parameter C, gamma, dan jenis kernel. Melalui *nested loops*, *source code* ini menginisialisasi dan melatih model SVM pada setiap kombinasi parameter yang telah ditentukan menggunakan SVC dari library *sklearn*. Performa model dievaluasi dan disimpan dalam matriks performa. Kemudian hasilnya disajikan sebagai serangkaian *heatmap* menggunakan *seaborn* dan *matplotlib*, menampilkan hasil dari performa SVM untuk setiap jenis kernel berdasarkan variasi parameter C dan gamma.

Untuk membantu analisis visual performa model terhadap kombinasi parameter yang berbeda, berikut visualisasi performa dari pemodelan yang telah didapatkan dengan menggunakan karnel *polynomial* terhadap nilai-nilai dari parameter C, dan juga nilai-nilai dari parameter *Gamma* dengan hasil visualisasi berikut:



Gambar 4. 17 Hasil visualisasi params

Dari visualisasi hasil *tunning* sentimen analisis dengan data teks cuitan dari media sosial Twitter pada Gambar 4.17 diatas dapat dibaca untuk hasil terbaik dengan berpatokan pada akurasi tertinggi yaitu yang ditepati pada kolom ketiga untuk C dengan nilai 10 dan baris keempat untuk gamma dengan nilai 0.1 kemudian dengan hasil yang paling tinggi didapat 0.84. setelah mendapatkan nilai-nilai tersebut, selanjutnya adalah melakukan penggambaran untuk visualisasi karnel yang telah didapat yaitu karnel *polynomial* dengan tampilan persebaran data serta dapat melihat pemisah antar data yang didapat atau *hyperplane*.

Berikut adalah *source code* visualisasi karnel *polynomial* dari tunning yang telah dilakukan model SVM dengan mencari *hyperplane*:

```
# Reduksi dimensi ke 2D
pca = PCA(n_components=2)
X_2d = pca.fit_transform(X_train_combined)

# Latih model SVM dengan parameter terbaik dari GridSearchCV
pada data yang sudah direduksi dimensi
best_svm = SVC(C=10, gamma=0.1, kernel='poly')
best_svm.fit(X_2d, y_train)

# Visualisasi hyperplane
xx, yy = np.meshgrid(np.linspace(X_2d[:, 0].min(), X_2d[:,
0].max(), 100),

np.linspace(X_2d[:, 1].min(), X_2d[:, 1].max(), 100))

Z = best_svm.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

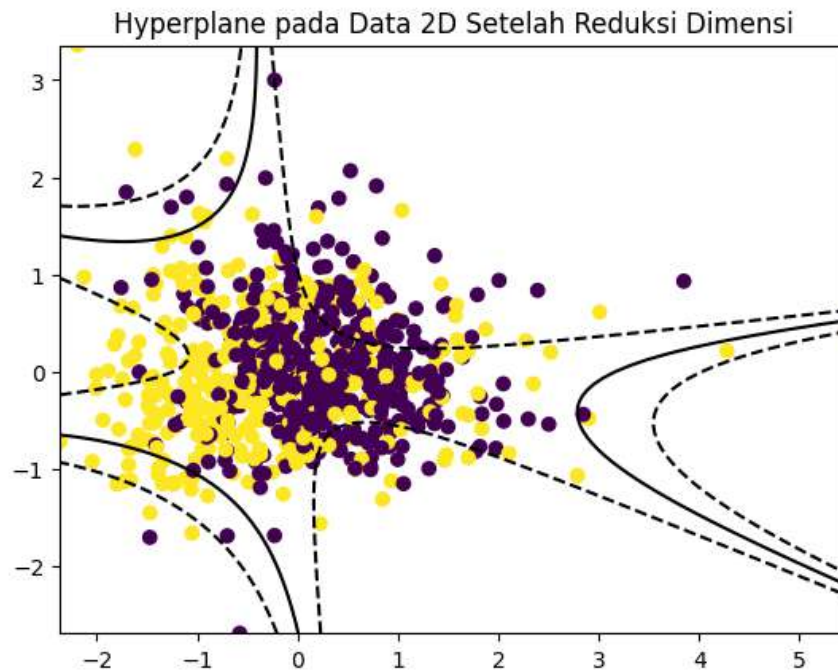
plt.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1],
linestyles=['--', '-', '--'])

plt.scatter(X_2d[:, 0], X_2d[:, 1], c=y_train, cmap='viridis')
plt.title("Hyperplane pada Data 2D Setelah Reduksi Dimensi")
plt.show()
```

Gambar 4.18 *Source code* visuliasasi *hyperplane*

Source code pada Gambar 5.18 di atas diterapkan dengan di atas menggunakan analisis PCA (*Principal Component Analysis*) untuk mereduksi dimensi fitur ke 2D, dikarenakan hasil dari *feature extraxtion* yang dihasilkan oleh *GloVe* adalah berdimensi 300D. Setelah reduksi, model *Support Vector Machine* (SVM) dilatih dengan parameter terbaik dari *GridSearchCV* menggunakan data yang sudah direduksi. Selanjutnya, kode memvisualisasikan *hyperplane* dari model SVM pada data 2D, menampilkan garis keputusan yang dibuat oleh model untuk memisahkan kelas, dengan warna dan pola garis yang merepresentasikan batas keputusan yang dihasilkan oleh model terhadap kelas-kelas pada data yang direduksi ke 2 dimensi.

Berikut adalah hasil visualisasi dengan menggunakan kernel *polynomial* dataset yang digunakan mempunyai karakter data yang tidak terstruktur kernel ini juga cocok untuk memecahkan masalah klasifikasi pada dataset pelatihan yang dinormalisasi sesuai dengan namanya, pada klasifikasi model ini adalah dengan mencari garis pemisah antar kelas dengan hasil berikut:



Gambar 4. 19 Hasil *hyperplane*

Pada Gambar 4.19 terdapat dua kelas yaitu kelas positif dan kelas negatif yang diwakili dalam kelas ungu adalah kelas positif dan kelas kuning adalah kelas negatif yang dipisahkan oleh garis *polynomial* berwarna hitam (*hyperplane*), kemudian area antar garis putus-putus adalah *margin* yang diperoleh berdasarkan jarak terdekat antara *hyperlane* dengan kelas yang ingin dipisahkan dan setiap kelas yang berperan sebagai penentu *margin* dikenal dengan istilah *support vector*. Dalam menentukan *hyperplane* metode *Support Vector Machine* (SVM) akan memilih margin yang paling besar *maximum margin* sebagai hasil akhir dengan hasil akhir kedua kelas masih terdapat sangat acak untuk persebaran datanya.

4.6 Validation

Pada langkah terakhir proses klasifikasi analisis sentimen ini adalah melakukan pengukuran dan evaluasi model menggunakan *confusion matrix*. *Confusion matrix* adalah sebuah tabel yang digunakan dalam pemahaman performa model klasifikasi pada *machine learning* dengan cara kerjanya menggambarkan jumlah prediksi *machine* yang benar terhadap data *real* dan jumlah prediksi *machine* yang salah terhadap data *real* yang dibuat oleh model terhadap data uji, memungkinkan untuk mengevaluasi sejauh mana model kita efektif dalam mengklasifikasikan data berdasarkan empat komponen utama yaitu TP (*True Positive*), TN (*True Negative*), FN (*False Negative*) dan FP (*False Positive*). Dari empat komponen tersebut, kita dapat menghitung berbagai metrik evaluasi model seperti akurasi, presisi, recall dan nilai F1-Score, yang memberikan pandangan yang lebih mendalam tentang kinerja model klasifikasi dalam mengatasi kasus analisis sentimen menggunakan proses *machine learning* pada studi kasus sentimen positif dan negatif. Berikut merupakan *source code* implementasi untuk *confusion matrix* beserta penjelasan hasilnya:

```

cm = confusion_matrix(y_test, prediction)

fig, ax = plt.subplots(figsize=(5,5))

ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.3)

for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(x=j, y=i, s=cm[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()

```

Gambar 4. 20 Source code confusion matrix

Source code pada Gambar 4.20 tersebut digunakan untuk menghasilkan dan menampilkan *confusion matrix* dari hasil kerja model *machine learning* terhadap data uji. *Confusion matrix* ini memberikan gambaran visual tentang seberapa baik model dapat mengklasifikasikan data menjadi kategori yang benar dan seberapa sering terjadi kesalahan prediksi, yang digunakan sebagai acuan atau untuk menentukan hasil dari perhitungan *accuracy*, *precision*, *Recall* dan *f1-score* dengan hasil *confusion matrix* sebagai berikut:

		Predicted	
		Positif	Negatif
Actual	Positif	93	16
	Negatif	16	73

Gambar 4. 21 Hasil tabel *confusion matrix*

Hasil tabel *confusion matrix* yang didapatkan pada Gambar 4.21 diatas dapat dijelaskan berdasarkan empat komponen utama yang didapatkan pada masing-masing kolom yaitu:

- TP (*True Positive*), sebanyak 93 data positif yang terklasifikasi benar.
- TN (*True Negative*), sebanyak 73 data negatif yang terklasifikasi benar.
- FP (*false positive*), yaitu sebanyak 16 data positif terklasifikasi salah.
- FN (*False Negative*), yaitu sebanyak 16 data negatif terklasifikasi salah.

Dari informasi tabel *confusin matrix* yang sudah didapat, selanjutnya adalah melakukan perhitungan *accuracy*, *preciission*, *Recall* dan *f1-score*. Untuk mengukur sejauh mana keberhasilan model dalam mengklasifikasikan data.

Accuracy (A) persentase prediksi benar dari *true positif* dan *true negative*.

$$A = \frac{(TP+TN)}{(TP+FP+FN+TN)} = \% \quad (4.11)$$

$$A = \frac{(93+73)}{(93+16+16+73)} = 0.83838383$$

$$Accuracy = 83\%$$

Precision (P) persentase prediksi benar dari dari seluruh nilai *positif*.

$$P = \frac{(TP)}{(TP+FP)} = \% \quad (4.12)$$

$$P = \frac{(93)}{(93+16)} = 0.85321100$$

$$Precision = 85\%$$

Recall (R) persentase prediksi *positif* dibandingkan dengan *true positif*.

$$R = \frac{(TP)}{(TP+FN)} = \% \quad (4.13)$$

$$R = \frac{(93)}{(93+16)} = 0.85321100$$

$$Recall = 85\%$$

F1-Score (F) merupakan perbandingan rata-rata *precision* dan *recall*.

$$F = \frac{2(P \times R)}{P+R} = \% \quad (4.14)$$

$$F = \frac{2(85.321101 \times 85.321101)}{85.321101 + 85.321101} = 0.85321100$$

$$F1 - Score = 85\%$$

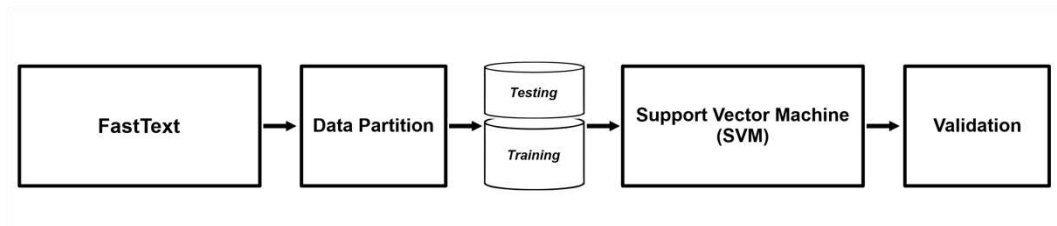
BAB V

ANALISIS SENTIMEN BERBASIS FASTTEXT DAN SVM

Bagian bab ini membahas tentang penerapan dan hasil pengujian kedua yaitu teknik *FastText* yang dipadukan dengan algoritma klasifikasi *Support Vector Machine* (SVM) dengan data yang sudah dipersiapkan pada tahap awal yaitu tahap *data preparation* sebelumnya.

5.1 Deskripsi Penelitian

Pemrosesan menggunakan *google colab* serta *python* sebagai bahasa pemrograman yang digunakan, *library tensorflow* untuk pembuatan model *machine learning* dan dataset teks sebagai bahan uji dengan alur berikut:



Gambar 5. 1 Analisis sentimen *FastText* dan SVM

Terlihat pada Gambar 5.1 alur pengujian pada analisis sentimen berbasis *FastText* dan SVM diatas akan dijelaskan secara umum pada setiap alur yang dilewati oleh data yang diproses. Untuk penjelasan secara umum adalah *FastText* mengekstrak data berbentuk teks dari proses *data preparation* sebelumnya kedalam nilai biner berbentuk vektor untuk dapat diproses pada algoritma SVM, kemudian data numrik berbentuk vektor tersebut dibagi menjadi dua bagian yaitu data *training* dan *testing*. Dengan tujuan data *taining* untuk melatih *machine* dan data *testing* untuk menguji keberhasilan *machine* dengan pengukuran validation *confusion matrix* berdasarkan nilai *accuracy*, *preccission*, *recall* dan *f1-score*, berikut merupakan langkah-langkah dan paparan hasil pengujian pada tahap kedua:

5.2 Import data dan Library

Langkah pertama adalah mempersiapkan data teks yang sudah melewati proses *data preparation* dan validasi, yang sudah benar-benar siap untuk dilakukan pemrosesan lebih lanjut yaitu klasifikasi teks analisis sentimen menggunakan teknik *FastText* pada tahap *feature extraction* dengan memasukan beberapa *library* yang dibutuhkan dan data dengan *source code* berikut:

```
import os
import csv
import nltk
import tweepy
!pip install fasttext
import fasttext
import numpy as np
import pandas as pd
nltk.download('punkt')
from tqdm.auto import tqdm
from nltk.corpus import stopwords
from gensim.models import FastText
from gensim.models import KeyedVectors
from nltk.tokenize import word_tokenize

df = pd.read_csv('/content/drive/MyDrive/Program Thesis/FASTTEXT
+ SVM/Data Bersih/Clean.csv')
df
```

Gambar 5. 2 Source code import library *FastText* dan import data

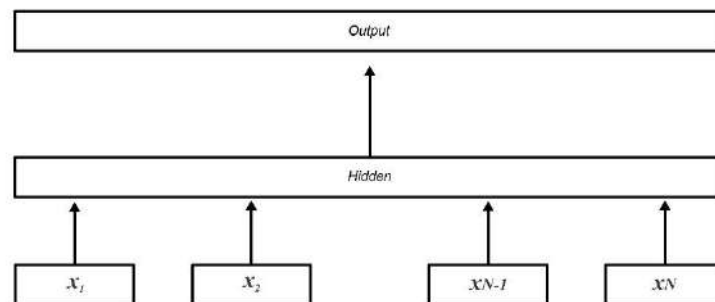
Source code pada Gambar 5.2 import beberapa *library* baris pertama hingga baris ke empat belas, kemudian mengambil *Clean.csv* dalam file komputer lokal yang diberi nama *df*, terdapat dua kolom yaitu label dan tweet dengan hasil berikut:

Tabel 5. 1 Hasil import data

No	label	tweet
0	0	paket tgl juni kirim cimahi semarang sampesamp...
1	1	siapsiap besok check out keranjang ya harbokir...
2	0	egk cok btw jnc tuh jincjancok sie ato
3	0	no kurir wilayah bogor selatan paket online am...
4	1	yaampun eonnie mmf mata ak buta

5.3 *FastText*

Merupakan library yang dikeluarkan oleh facebook yang dapat diterapkan pada teknik *feature extraction*. Sebenarnya, *FastText* sendiri adalah pengembangan dari library *Word2Vec* yang telah lebih lama terkenal dan sering digunakan. *FastText* memiliki keunggulan dibanding *Word2Vec*, satunya adalah kemampuan *FastText* untuk menangani kata yang tidak pernah dijumpai sebelumnya *Out of Vocabulary* atau dikenal OOV yang dibentuk menggunakan *Sub-Word*. Misalnya kata-kata yang tidak baku seperti “Pengoptimalisasian” tetap akan diperoleh vektornya dengan memecah kata tersebut menjadi beberapa bagian “peng”, “optima”, “lisa”, “sian”. Sedangkan kakaknya *Word2Vec* hanya mampu melihat kata disekitarnya secara makna (*Semantic*) dan akan menghasilkan eror ketika menerima kata yang tidak pernah ada di kamus OOV, berikut merupakan gambaran alur pemrosesan pada *feature extraction* untuk teknik *FastText*.



Gambar 5.3 *Layer fasttext*

Dari Gambar 5.3 merupakan gambaran model arsitektur dari teknik *FastText* terdapat tiga layer, untuk layer dasar dengan kalimat dengan fitur N yaitu ngram x_1, x_2, \dots, x_n dengan fitur fitur yang disematkan guna membentuk variabel tersembunyi pada layer *hidden* agar menghasilkan keluaran angka berbentuk vektor yaitu pada layer *output* dengan langkah-langkah detail sebagai berikut:

Langkah pertama adalah representasi *bag-of-words* dari teks pertama dimasukkan ke dalam *lookup layer*, di mana embeddings diambil setiap kata dengan *source code* dan beserta hasilnya berikut:

```

responsFastText
fasttext.train_unsupervised('/content/drive/MyDrive/Program
Thesis/FASTTEXT + SVM/Data Bersih/clean.txt', model="skipgram",
dim=256, lr=0.01, epoch=1000, minCount=1)

fst = responsFastText

fst["wkwk"]

```

Gambar 5. 4 Source code representasi *bag of word*

Source code pada Gambar 5.4 merupakan perintah menggunakan library *FastText* untuk melatih model *unsupervised* menggunakan file teks '*clean.txt*'. Dengan parameter seperti model "*skipgram*", dimensi vektor 256, laju pembelajaran 0.01, 1000 epoch dan minimum count 1. Kemudian, deretan matrix tersebut disimpan kedalam suatu ekstensi bernama *.bin* serta pengujian representasi vektor kata "*wkwk*" setelah proses pelatihan model selesai untuk mengetahui hasil kata yang tidak pernah ada dalam kamus yang sudah dilatih, dengan hasil berikut:

```

array([-0.5973275 , -0.11672792, -0.67471075,  0.2745443 ,  0.05749161,
        0.01739001,  0.04956616,  0.60788196,  0.9957117 ,  0.87569517,
        0.7510198 ,  0.75292224, -0.32306308,  0.51762044, -0.33856717,
       -0.21553124, -0.6688066 ,  0.03667931, -0.5491794 ,  0.37162825,
       -0.2903049 , -0.37966624, -0.30382922,  0.4029754 ,  0.71958244,
       -0.06121159,  0.5358847 ,  0.0577746 ,  0.05569059,  0.42941797,
       -0.2708197 ,  0.13051742,  0.37914252,  0.2204586 , -0.05816022,
       -0.20957468,  0.10293452, -0.9961912 , -0.0375546 , -0.21849187,
        0.23363513,  0.12901059,  0.516939 , -0.22265935, -0.06020863,
       -0.25750256,  0.12895806,  0.07803058, -0.42183396,  0.24921529,
       -0.40305308,  0.63154924, -0.27249834,  0.17427324, -0.08124056,
       -0.5751028 , -0.35806835,  0.20581014,  0.09270946,  0.66609865,
       -0.30479813, -0.38562918, -0.22159313, -0.03857971, -0.28414467,
       -0.61576444, -0.30974463,  0.3723034 ,  1.143081 ,  0.28707263,
       -0.4085266 , -0.19759454, -0.7572483 ,  0.27084285, -0.3594067 ,
       -0.68849623,  0.23090132,  0.24828628, -0.6872239 , -0.12945858,
       -0.19253571,  0.20121092,  0.244353 , -0.27493086,  0.23533742,
       -0.4227166 , -0.07774404,  0.5530928 ,  0.18886764,  0.18013732,

```

```

-0.40581518, -0.14140931, 0.41596285, -0.68948674, -0.00960646,
-0.11277815, 0.31105486, -0.4207177, 0.349322, 0.40137005,
-0.42467982, 0.22120565, 0.28231, -0.7813899, -0.75542706,
-0.04356801, -0.3270332, 0.25977027, 0.2248216, -0.19758223,
-0.38711852, -0.08926409, -0.14186883, 0.03814009, -0.39425388,
0.34024677, -0.10780679, 0.52004576, 0.04368182, 0.07115758,
0.25983182, -0.02491902, -0.04196392, 0.23671177, -0.5577886,
-0.81810457, -0.02911445, -0.7813794, 0.27565458, -0.799331,
-0.08470254, 0.14614788, 0.16770363, -0.08028569, -0.42479208,
0.4451428, -0.13081992, -0.0610542, 0.15989201, 0.33333954,
-0.11253244, -0.20590924, -0.59507704, -0.08003447, -0.04752825,
0.19652121, 0.95099765, 0.01032178, 0.16004474, -0.28702483,
-0.2883537, 0.13754393, -0.39128608, 0.77463216, -0.12657009,
-0.11936058, -0.4060192, 0.2242946, 0.5108794, -0.22156549,
0.1958638, -0.32370377, 0.40855727, -0.6450719, 0.44985795,
-0.4676319, 0.3160115, -0.17269234, 0.05771348, -0.43846926,
0.5943387, 0.5371973, -0.19873822, -0.07053103, 0.01473887,
0.17574067, 0.22231238, 0.48596764, 0.48232344, 0.99171036,
-0.37054163, -0.7212385, -0.253014, -0.15924722, -0.11843912,
0.01644303, -0.91949415, 0.17598543, -0.92697734, 0.41088662,
-0.13849068, 0.68958586, -0.02771055, -0.00203554, 0.2410119,
0.43443018, -0.6092401, -0.12103393, -0.22591949, -0.1244894,
-0.12963271, 0.05562657, 0.7763889, 0.1146599, -0.07636209,
0.08051109, -0.74456954, 0.02146261, 0.23098537, -0.05819391,
-0.1713523, 0.07222203, -0.29427853, -0.6086546, 0.06355329,
-0.4202356, -0.42560673, 0.06092966, 0.52949077, 0.5369301,
0.4229154, -0.2730333, 0.5998388, -0.06648129, 0.2916918,
0.22024286, 0.22819921, -0.54041225, -0.00451532, -0.11283201,
-0.32816792, 0.66597396, -0.20387714, -0.2114055, -0.96426725,
0.0772072, -0.8889966, 0.515992, -0.01585408, 0.12184266,
0.102027, 0.48267555, 0.84512615, 0.529688, 0.65876335,
-0.07373106, -0.61985266, 0.17978655, 0.32920924, 0.35592118,
0.0433934, 1.3735415, -0.5077087, -0.7849762, -0.39470688,
-0.11893404], dtype=float32)

```

Gambar 5. 5 Hasil representasi *bag of word*

Langkah kedua adalah rekonstruksi data setelah semua kata mendapatkan nilai vektor dengan melalui proses yang sudah diterapkan kemudian langkah selanjutnya adalah penyisipan kata, kata tersebut dirata-ratakan sehingga diperoleh satu penyematan rata-rata untuk keseluruhan teks atau dokumen yang telah dilakukan pemrosesan. Pada *hidden layer* kita berakhir dengan $n_words \times dim$, di mana *dim* adalah ukuran embeddings dan n_words adalah ukuran kosakata. Setelah rata-rata, selanjutnya kita hanya memiliki satu vektor yang kemudian diproses ke pengklasifikasian yang diterapkan dengan *source code* berikut:

```

fst      =      fasttext.load_model("/content/drive/MyDrive/Program
Thesis/FASTTEXT + SVM/HasilFastText/Model.bin")

def norm_sent_vector(sentence, fst_model, stopwords):
    vecs = [fst_model.get_word_vector(word.lower()) for word in
word_tokenize(sentence) if word.lower() not in stopwords]
    norm_vecs = [vec / np.linalg.norm(vec) for vec in vecs if vec
is not None and np.linalg.norm(vec) > 0]
    sent_vec = np.mean(norm_vecs, axis=0)
    return sent_vec

nltk.download('stopwords')
stopwords_indonesia = stopwords.words('indonesian')

```

Gambar 5. 6 Source code rekontruksi data FastText

Source code pada Gambar 5.6 pada baris pertama adalah mengambil data yang sudah disimpan pada proses sebelumnya dengan nama 'Model.bin' dengan berisikan kumpulan matrik setiap kata pada sebuah dokumen, kemudian Fungsi 'norm_sent_vector' menghitung vektor normalisasi rata-rata dari kalimat, mengabaikan kata-kata tertentu, dan menggunakan model kata-kunci untuk menghasilkan vektor kata yang dinormalisasi.

Langkah ketiga selanjutnya menerapkan *softmax* pada transformasi linier dari *layer output* dan *layer input*. Transformasi linier adalah matriks dengan *dim x n_output*, di mana *n_output* adalah jumlah kelas, dengan persamaan berikut:

$$-\frac{1}{N} \sum_{n=1}^N y_n \log(f(BAx_n)) \quad (5.1)$$

- x_n adalah representasi *one-hot-encoded* dari sebuah kata (*n-gram feature*)
- A adalah *look-up matrix* yang mengambil kata embedding
- B adalah transformasi dari *linier output*
- f adalah *function* dari *softmax*

Dari persamaan 5.1 dapat dilakukan pemrosesan dengan *source code* berikut:

```

vecs = [norm_sent_vector(sentence, fst, stopwords_indonesia) for
sentence in df.tweet]
vecs = np.array(vecs)

vecs
vecs.shape

```

Gambar 5. 7 Source code penerapan softmax

Dengan menjalankan *source code* pada Gambar 5.7 akan mendapatkan matriks NumPy (*vecs*) yang berisi vektor normalisasi untuk setiap kalimat dalam DataFrame *df*, yang menghasilkan matriks akhir berikut:

```

array([[ -0.03408794, -0.00706709, -0.00084954, ..., -0.05022984,
         0.03081832,  0.02187653],
       [ -0.047764  ,  0.0263945  , -0.03243399, ..., -0.03404212,
        -0.0324075  , -0.03316876],
       [  0.00072248,  0.07202232, -0.04152314, ..., -0.10147168,
         0.01417872, -0.04323873],
       ...,
       [ -0.07497261,  0.01123452, -0.04434814, ..., -0.01351779,
        -0.01964731,  0.0331611  ],
       [  0.02681901,  0.01004733, -0.02492669, ..., -0.00321598,
         0.03777692,  0.02607031],
       [ -0.07518149, -0.01218531, -0.04864071, ..., -0.02180763,
        -0.05297254,  0.0432808  ]], dtype=float32)

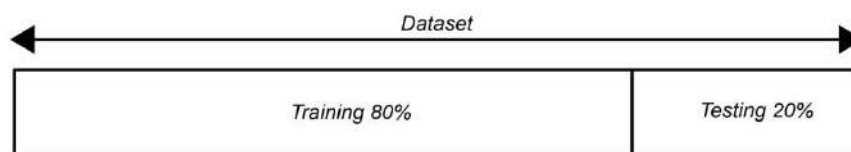
```

Gambar 5. 8 Hasil akhir matrik

Kemudian untuk hasil pada Gambar 5.8 merupakan vektor yang didapatkan adalah dengan menampilkan bentuk (jumlah baris dan kolom) dari array vektor yang telah didapatkan sebelumnya. Hal ini memberikan informasi tentang seberapa banyak kalimat yang telah diolah dan berapa panjang vektor normalisasi untuk setiap kalimat dengan hasil 986 kalimat, serta mempunyai 256 panjang dimensi vektor yang didapatkan oleh model *FastText* yang dilakukan. Proses ini memungkinkan pemahaman yang lebih mendalam tentang representasi vektor yang dihasilkan oleh model *FastText* dan bagaimana informasi ini dapat diterapkan dalam konteks klasifikasi analisis sentimen lebih lanjut yaitu dengan melanjutkan pada pengolahan modeling atau implementasi metode SVM (*Support Vector Machine*).

5.3 Data Partition

Langkah kedua adalah membagi data yang sudah bersih dan berlabel tersebut kedalam dua bagian yaitu data *training* dan data *testing*, pada pembagian data ini peneliti menggunakan aturan umum (*Rule of Thumb*) dalam persentase yaitu 80% digunakan sebagai data *training* dan 20% digunakan sebagai data *testing* dengan ilustrasi pada gambar dibawah:



Gambar 5. 9 Pembagian *training data* dan *testing data*

Training bagian dataset yang digunakan untuk melatih dan membuat prediksi atau menjalankan fungsi dari sebuah algoritma klasifikasi serta sebagai sumber data mesin yang kita latih untuk mencari korelasi pembelajaran pada pola data tersebut, sedangkan *Testing* merupakan bagian dataset yang digunakan untuk menguji dan melihat keakuratan *machine* yang sudah melakukan proses pembelajaran dan pengenalan pola yang sudah dilakukan pada data *training* dengan *source code*:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(vecs,
df['label'], test_size = 0.20, random_state = 42)

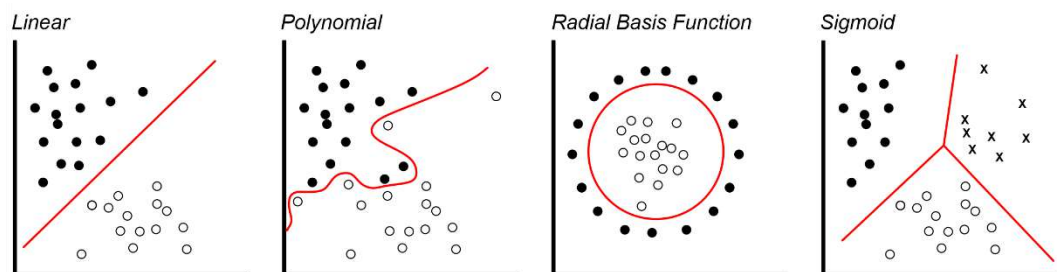
X_train.shape
X_test.shape
```

Gambar 5. 10 *Source code* split data *training* dan *testing*

Source code pada Gambar 5.10 pembagian vektor fitur (*vecs*) dan label (*df['label']*) dengan proporsi tertentu yaitu 80%:20%, Dengan hasil akhir pembagian yang didapatkan yaitu sebanyak 788 data *training* dan 198 sebagai data *testing* dengan perbandingan 80%:20%, serta Panjang dari vektornya adalah 256.

5.5 Support Vector Machine (SVM)

Setelah data melewati proses *splitting* dan proses *feature extraction*, saat ini data menjadi bentuk numerik berupa vektor lalu pada tahap inilah dilakukan proses modeling klasifikasi dengan menerapkan metode *Support Vector Machine* (SVM). Algoritma pembelajaran ini pertama kali dikembangkan oleh Boser, Guyon dan Vapnik pada tahun 1992 yang merupakan sistem pembelajaran dengan menggunakan hipotesis berupa fungsi-fungsi linier didalam sebuah fitur yang memiliki dimensi tinggi dan dilatih menggunakan teori optimasi.



Gambar 5. 11 Algoritma *Support Vector Machine* (SVM)

Pada dasarnya konsep dan cara kerja dari metode *Support Vector Machine* (SVM) adalah berusaha mencari dan menemukan fungsi garis pemisah (*hyperplane*) yang terbaik diantara beberapa fungsi. Seperti pada Gambar 5.11 terdapat 4 fungsi atau karnel dengan tipe data dan pola masing-masing dan masing-masing memiliki jenis garis pemisah yang berbeda-beda pada setiap fungsinya. Metode *Support Vector Machine* (SVM) mencari dan memilih dari salah-satu garis/*hyperplane* yang paling baik dan pemilihan tersebut sangat bergantung pada jenis data dengan tujuan dapat memisahkan antar kelas yang berbeda dengan sempurna. Formulasi algoritma ini dapat dibedakan menjadi beberapa fungsi atau karnel yang digunakan yaitu *linear*, *polynomial*, *radial basis function* dan *sigmoid* dengan cara mencari dan mengoptimalkan pemrosesan yaitu proses *tunning*.

Langkah awal adalah *tuning Support Vector Machine (SVM)* dalam *machine learning* adalah proses untuk mengoptimalkan parameter-parameter model SVM agar memberikan kinerja yang terbaik untuk tugas yang sedang dihadapi. Berikut adalah langkah-langkah dalam proses tuning SVM dalam penerapan *source code*:

```
param_grid = {'C': [0.1, 1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
              'kernel': ['rbf', 'linear', 'poly', 'sigmoid']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

grid.fit(X_train_combined, y_train)
```

Gambar 5. 12 *Source code tuning SVM*

- d. Pemilihan karnel, memilih jenis kernel function yang akan digunakan untuk menentukan cara SVM akan memetakan data ke ruang fitur yang lebih tinggi serta yang digunakan *linear*, *polynomial*, *Radial Basis Function (RBF)* dan *sigmoid*. Pemilihan kernel sangat berpengaruh besar pada hasil model didapat.
- e. Penentuan parameter C, dalam mengatur *trade-off* antara margin yang lebih besar dan kesalahan klasifikasi yang lebih kecil. Nilai C yang lebih besar akan memberikan margin yang lebih kecil tetapi kesalahan klasifikasi yang lebih kecil, sementara nilai C yang lebih kecil akan memberikan margin yang lebih besar tetapi kesalahan klasifikasi yang lebih besar. Ini adalah langkah penting dalam tuning SVM, dengan metode *GridSearch* untuk menemukan nilai terbai
- f. Penentuan parameter gamma, Parameter ini mengontrol fleksibilitas model terhadap data pelatihan. Dengan berpatokan pada hasil dengan gambaran nilai gamma yang lebih besar dapat menghasilkan model yang lebih rumit, sementara nilai gamma yang lebih kecil dapat menghasilkan model yang lebih sederhana. Sama seperti parameter C, serta dapat mencoba berbagai nilai gamma untuk menemukan hasil yang optimal.

Berikut merupakan hasil tuning yang dilakukan *machine* terhadap data yang telah diproses dengan berbagai parameter yang telah di masukan sebagai pilihan untuk mencari model SVM yang paling optimal:

```

Fitting 5 folds for each of 100 candidates, totalling 500 fits
[CV 1/5] END .....C=10, gamma=0.01, kernel=poly;; score=0.557 total time= 0.1s
[CV 2/5] END .....C=10, gamma=0.01, kernel=poly;; score=0.557 total time= 0.1s
[CV 3/5] END .....C=10, gamma=0.01, kernel=poly;; score=0.557 total time= 0.1s
[CV 4/5] END .....C=10, gamma=0.01, kernel=poly;; score=0.561 total time= 0.0s
[CV 5/5] END .....C=10, gamma=0.01, kernel=poly;; score=0.561 total time= 0.1s
[CV 1/5] END ..C=10, gamma=0.01, kernel=sigmoid;; score=0.778 total time= 0.1s
[CV 2/5] END ..C=10, gamma=0.01, kernel=sigmoid;; score=0.766 total time= 0.1s
[CV 3/5] END ..C=10, gamma=0.01, kernel=sigmoid;; score=0.703 total time= 0.1s
[CV 4/5] END ..C=10, gamma=0.01, kernel=sigmoid;; score=0.764 total time= 0.1s
[CV 5/5] END ..C=10, gamma=0.01, kernel=sigmoid;; score=0.758 total time= 0.1s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;; score=0.557 total time= 0.1s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;; score=0.557 total time= 0.1s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;; score=0.557 total time= 0.1s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;; score=0.561 total time= 0.1s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;; score=0.561 total time= 0.1s
[CV 1/5] END ..C=10, gamma=0.001, kernel=linear;; score=0.823 total time= 0.0s
[CV 2/5] END ..C=10, gamma=0.001, kernel=linear;; score=0.835 total time= 0.0s
[CV 3/5] END ..C=10, gamma=0.001, kernel=linear;; score=0.747 total time= 0.0s
[CV 4/5] END ..C=10, gamma=0.001, kernel=linear;; score=0.841 total time= 0.0s
[CV 5/5] END ..C=10, gamma=0.001, kernel=linear;; score=0.809 total time= 0.0s
[CV 1/5] END ....C=10, gamma=0.001, kernel=poly;; score=0.557 total time= 0.1s
[CV 2/5] END ....C=10, gamma=0.001, kernel=poly;; score=0.557 total time= 0.1s
[CV 3/5] END ....C=10, gamma=0.001, kernel=poly;; score=0.557 total time= 0.1s
[CV 4/5] END ....C=10, gamma=0.001, kernel=poly;; score=0.561 total time= 0.1s
[CV 5/5] END ....C=10, gamma=0.001, kernel=poly;; score=0.561 total time= 0.1s
[CV 1/5] END .C=10, gamma=0.001, kernel=sigmoid;; score=0.557 total time= 0.1s
[CV 2/5] END .C=10, gamma=0.001, kernel=sigmoid;; score=0.557 total time= 0.1s
[CV 3/5] END .C=10, gamma=0.001, kernel=sigmoid;; score=0.557 total time= 0.1s
[CV 4/5] END .C=10, gamma=0.001, kernel=sigmoid;; score=0.561 total time= 0.1s
[CV 5/5] END .C=10, gamma=0.001, kernel=sigmoid;; score=0.561 total time= 0.1s
[CV 1/5] END ....C=10, gamma=0.0001, kernel=rbf;; score=0.557 total time= 0.1s
[CV 2/5] END ....C=10, gamma=0.0001, kernel=rbf;; score=0.557 total time= 0.1s
[CV 3/5] END ....C=10, gamma=0.0001, kernel=rbf;; score=0.557 total time= 0.1s
[CV 4/5] END ....C=10, gamma=0.0001, kernel=rbf;; score=0.561 total time= 0.1s
[CV 5/5] END ....C=10, gamma=0.0001, kernel=rbf;; score=0.561 total time= 0.1s
[CV 1/5] END .C=10, gamma=0.0001, kernel=linear;; score=0.823 total time= 0.0s
[CV 2/5] END .C=10, gamma=0.0001, kernel=linear;; score=0.835 total time= 0.0s
[CV 3/5] END .C=10, gamma=0.0001, kernel=linear;; score=0.747 total time= 0.0s
[CV 4/5] END .C=10, gamma=0.0001, kernel=linear;; score=0.841 total time= 0.0s
[CV 5/5] END .C=10, gamma=0.0001, kernel=linear;; score=0.809 total time= 0.0s
[CV 1/5] END ...C=10, gamma=0.0001, kernel=poly;; score=0.557 total time= 0.1s
[CV 2/5] END ...C=10, gamma=0.0001, kernel=poly;; score=0.557 total time= 0.1s
[CV 3/5] END ...C=10, gamma=0.0001, kernel=poly;; score=0.557 total time= 0.1s
[CV 4/5] END ...C=10, gamma=0.0001, kernel=poly;; score=0.561 total time= 0.1s
[CV 5/5] END ...C=10, gamma=0.0001, kernel=poly;; score=0.561 total time= 0.1s
[CV 1/5] END C=10, gamma=0.0001, kernel=sigmoid;; score=0.557 total time= 0.1s
[CV 2/5] END C=10, gamma=0.0001, kernel=sigmoid;; score=0.557 total time= 0.1s
[CV 3/5] END C=10, gamma=0.0001, kernel=sigmoid;; score=0.557 total time= 0.1s
[CV 4/5] END C=10, gamma=0.0001, kernel=sigmoid;; score=0.561 total time= 0.1s
[CV 5/5] END C=10, gamma=0.0001, kernel=sigmoid;; score=0.561 total time= 0.1s
[CV 1/5] END .....C=100, gamma=1, kernel=rbf;; score=0.791 total time= 0.1s
[CV 2/5] END .....C=100, gamma=1, kernel=rbf;; score=0.867 total time= 0.1s
[CV 3/5] END .....C=100, gamma=1, kernel=rbf;; score=0.766 total time= 0.1s
[CV 4/5] END .....C=100, gamma=1, kernel=rbf;; score=0.841 total time= 0.1s
[CV 5/5] END .....C=100, gamma=1, kernel=rbf;; score=0.828 total time= 0.1s
[CV 1/5] END ....C=100, gamma=1, kernel=linear;; score=0.778 total time= 0.1s
[CV 2/5] END ....C=100, gamma=1, kernel=linear;; score=0.829 total time= 0.1s
[CV 3/5] END ....C=100, gamma=1, kernel=linear;; score=0.759 total time= 0.1s
[CV 4/5] END ....C=100, gamma=1, kernel=linear;; score=0.815 total time= 0.1s
[CV 5/5] END ....C=100, gamma=1, kernel=linear;; score=0.783 total time= 0.1s
[CV 1/5] END .....C=100, gamma=1, kernel=poly;; score=0.804 total time= 0.1s
[CV 2/5] END .....C=100, gamma=1, kernel=poly;; score=0.886 total time= 0.0s
[CV 3/5] END .....C=100, gamma=1, kernel=poly;; score=0.766 total time= 0.1s

```


[CV 2/5] END C=100, gamma=0.0001, kernel=linear;;	score=0.829	total time=	0.1s
[CV 3/5] END C=100, gamma=0.0001, kernel=linear;;	score=0.759	total time=	0.0s
[CV 4/5] END C=100, gamma=0.0001, kernel=linear;;	score=0.815	total time=	0.0s
[CV 5/5] END C=100, gamma=0.0001, kernel=linear;;	score=0.783	total time=	0.0s
[CV 1/5] END ..C=100, gamma=0.0001, kernel=poly;;	score=0.557	total time=	0.0s
[CV 2/5] END ..C=100, gamma=0.0001, kernel=poly;;	score=0.557	total time=	0.1s
[CV 3/5] END ..C=100, gamma=0.0001, kernel=poly;;	score=0.557	total time=	0.0s
[CV 4/5] END ..C=100, gamma=0.0001, kernel=poly;;	score=0.561	total time=	0.1s
[CV 5/5] END ..C=100, gamma=0.0001, kernel=poly;;	score=0.561	total time=	0.0s
[CV 1/5] ENDC=1000, gamma=1, kernel=rbf;;	score=0.791	total time=	0.1s
[CV 2/5] ENDC=1000, gamma=1, kernel=rbf;;	score=0.867	total time=	0.1s
[CV 3/5] ENDC=1000, gamma=1, kernel=rbf;;	score=0.766	total time=	0.0s
[CV 4/5] ENDC=1000, gamma=1, kernel=rbf;;	score=0.841	total time=	0.1s
[CV 5/5] ENDC=1000, gamma=1, kernel=rbf;;	score=0.828	total time=	0.1s
[CV 1/5] END ...C=1000, gamma=1, kernel=linear;;	score=0.785	total time=	0.3s
[CV 2/5] END ...C=1000, gamma=1, kernel=linear;;	score=0.804	total time=	0.2s
[CV 3/5] END ...C=1000, gamma=1, kernel=linear;;	score=0.759	total time=	0.2s
[CV 4/5] END ...C=1000, gamma=1, kernel=linear;;	score=0.841	total time=	0.2s
[CV 5/5] END ...C=1000, gamma=1, kernel=linear;;	score=0.828	total time=	0.3s
[CV 1/5] ENDC=1000, gamma=1, kernel=poly;;	score=0.816	total time=	0.0s
[CV 2/5] ENDC=1000, gamma=1, kernel=poly;;	score=0.886	total time=	0.0s
[CV 3/5] ENDC=1000, gamma=1, kernel=poly;;	score=0.766	total time=	0.0s
[CV 4/5] ENDC=1000, gamma=1, kernel=poly;;	score=0.847	total time=	0.0s
[CV 5/5] ENDC=1000, gamma=1, kernel=poly;;	score=0.822	total time=	0.0s

Gambar 5. 13 Hasil *tunning* SVM

Pada penerapan model SVM yang digunakan dan telah dilakukan *tunning* dengan hasil pemrosesan *machine* seperti pada Gambar 5.13 telah didapatkan hasil yang paling optimal pada nilai $C=10$, $\gamma=1$ dan kernel=polynomial. Dengan cara mencari keluaran hasil pencarian terbaik menggunakan *grid.best_params_*.

```
print(grid.best_params_)
```

Gambar 5. 14 Source code show best *tunning*

Dengan hasil output berikut:

```
{'C': 10, 'gamma': 1, 'kernel': 'poly'}
```

Gambar 5. 15 Output best *tunning*

Setelah memperoleh hasil *tuning* untuk model yang telah ditetapkan yaitu *Support Vector Machine* (SVM), langkah berikutnya adalah menampilkan hasil nilai visual berupa grafik berbentuk tabel dengan melibatkan pemahaman visual yang mendalam terhadap model yang telah disusun. Ini mencakup analisis nilai C , evaluasi hasil nilai γ , dan pemahaman terperinci terkait jenis kernel yang digunakan pada parameter ini dapat memahami dampaknya terhadap kinerja keseluruhan model *Support Vector Machine* (SVM).

Pada dasarnya, *Support Vector Machine* SVM merupakan suatu *linear classifier*. Namun, SVM dapat dikembangkan menjadi *nonlinear classifier* dengan konsep kernel *trick* pada ruang berdimensi lebih tinggi dapat menangani *nonlinear classifier*. terdapat fungsi kernel *trick* pada SVM yang sudah didapatkan pada proses *tunning* sebelumnya adalah karnel *polynomial* dengan persamaan berikut:

$$K(x_i, x_j) = ((x_i, x_j) + 1)^d \quad (5.2)$$

Dengan:

K = Fungsi karnel

x_i, x_j = Fektor dari dataset

d = Pangkat *polynomial*

Kemudian untuk langkah-langkah persamaan secara umum menggunakan metode *Support Vector Machine* (SVM) sebagai berikut:

1. Merepresentasikan kata-kata dalam bentuk vektor numerik berdasarkan distribusi kata-kata cuitan twitter dalam konteks korpus teks *GloVe*;
2. Menginisialisasi berbagai parameter yang diperlukan dalam perhitungan manual dengan SVM seperti a_i , γ , C , s , λ , dan i_{max} ;

Keterangan:

a_i = Alpha atau *lagrange multiplier* untuk mencari *support vector*

γ = *Learning rate* untuk mengontrol kecepatan

C = *Cost* untuk meminimalkan nilai *error* saat proses *training*

s = Epsilon untuk ukuran tingkat *error* klasifikasi

λ = Turunan batas teoritis

i_{max} = Iterasi maksimum

3. Melakukan perhitungan kernel *polynomial* terlebih dahulu;
4. Melakukan perhitungan terhadap nilai y atau label dari kelas yang telah ditentukan yaitu kelas positif (1) dan kelas negatif(0);
5. Menghitung matriks *Hessian* dengan menggunakan persamaan berikut:

$$D_{ij} = y_i y_j (K(x_i, x_j) + \lambda^2) \quad (5.3)$$

Dengan nilai i dan $j = 1, 2, 3, \dots, n$.

Keterangan:

D = matriks *Hessian*

y_i = kelas data ke- i

y_j = kelas data ke- j

$K(x_i, x_j)$ = fungsi kernel yang digunakan

6. Menghitung nilai *error rate* dengan tujuan untuk mencerminkan seberapa sering model membuat kesalahan dalam memprediksi kelas atau nilai target.

$$E_i = \sum_{j=1}^n a_j D_{ij} \quad (5.4)$$

Keterangan:

E_i = nilai *error rate* data ke- i

7. Menghitung nilai delta alpha untuk melihat perubahan fungsi yang disebut dengan *Lagrange Multiplier*. Jika data *training* telah mencapai nilai konvergen ($\max(|\delta a_i|) < \mathcal{E}$) dan ketika maksimum iterasi mencapai nilai yang ditentukan, maka iterasi akan berhenti;

$$\delta a_i = \min(\max[\gamma(1 - E_i), a_i], C - a_i) \quad (5.5)$$

Keterangan:

δa_i = nilai delta alfa data ke- i

8. Menghitung nilai a_i baru, koefisien yang baru dihitung sebagai bagian dari vektor bobot (w) yang menggambarkan *hyperplane* pemisah agar dapat digunakan pada iterasi selanjutnya;

$$a_i = a_i + \delta a_i \quad (5.6)$$

9. Menghitung nilai $w \cdot x^+$ dan $w \cdot x^-$ untuk mendapatkan nilai bias b atau parameter yang bertanggung jawab untuk menentukan posisi *hyperplane*;

$$w \cdot x^+ = a_i y_i K(x_i, x^+) \quad (5.7)$$

$$w \cdot x^- = a_i y_i K(x_i, x^-) \quad (5.8)$$

$$b = -\frac{1}{2} (w \cdot x^+ + w \cdot x^-) \quad (5.9)$$

Keterangan:

$w \cdot x^+$ = nilai kernel data x dengan data x kelas positif

$w \cdot x^-$ = nilai kernel data x dengan data x kelas negatif

b = nilai bias

10. Setelah mendapatkan nilai bias, lalu menghitung kernel dan nilai bobot dari data *testing* agar dapat dimasukkan ke dalam fungsi klasifikasi dari kedua data *testing* untuk menentukan kelas data uji;

$$f(x) = \text{sign} \sum_{i=0}^n (a_i y_i K(x, x_i) + b) \quad (5.10)$$

Keterangan:

w = parameter *hyperplane* yang dicari garis tegak lurus antara garis *hyperplane* dan titik *support vector*

x = titik data masukan *Support Vector Machine*

a_i = nilai bias

$K(x, x_i)$ = fungsi kernel

Berikut adalah *source code* visualisasi tabel hasil dari tuning yang telah dilakukan model SVM dengan mencari hasil nilai yang paling tinggi:

```
# Definisikan parameter yang ingin Anda jelajahi
C_range = [0.1, 1, 10, 100, 1000]
gamma_range = [0.0001, 0.001, 0.01, 0.1, 1]
kernels = ['rbf', 'linear', 'poly', 'sigmoid']

# Inisialisasi array untuk menyimpan metrik performa
performa = np.zeros((len(C_range), len(gamma_range),
len(kernels)))

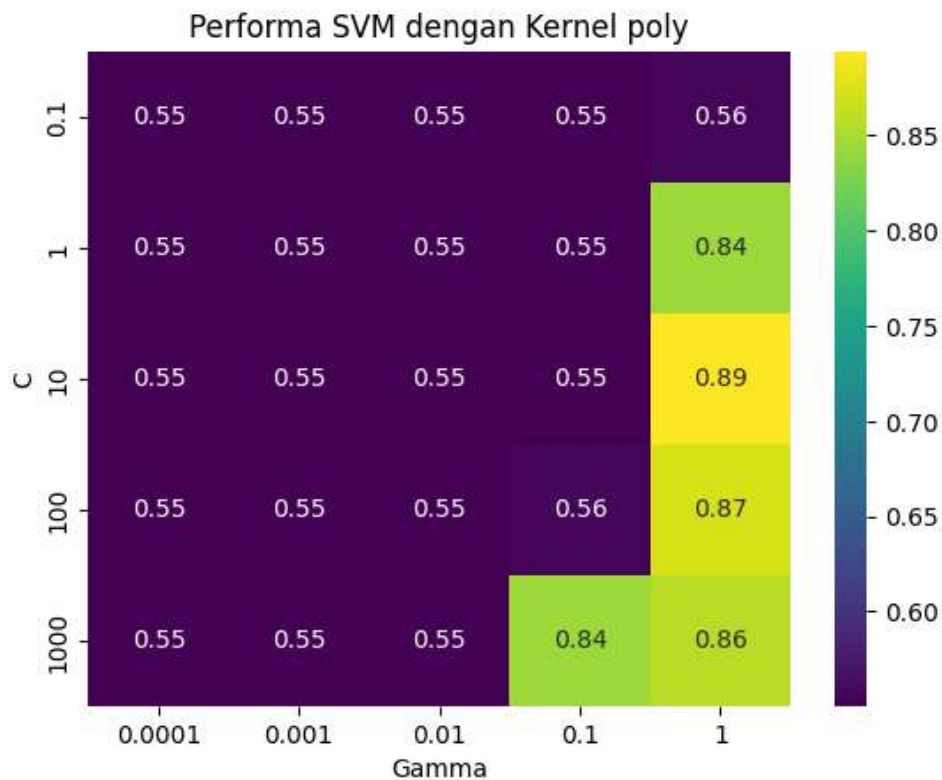
# Loop untuk mengeksekusi GridSearchCV pada setiap kombinasi
parameter
for i, C in enumerate(C_range):
    for j, gamma in enumerate(gamma_range):
        for k, kernel in enumerate(kernels):
            svc = SVC(C=C, gamma=gamma, kernel=kernel)
            svc.fit(X_train, y_train)
            performa[i, j, k] = svc.score(X_test, y_test)

# Buat heatmap untuk setiap kernel
for k, kernel in enumerate(kernels):
    plt.figure()
    sns.heatmap(performa[:, :, k], annot=True,
xticklabels=gamma_range, yticklabels=C_range, cmap='viridis')
    plt.title(f'Performa SVM dengan Kernel {kernel}')
    plt.xlabel('Gamma')
    plt.ylabel('C')
    plt.show()
```

Gambar 5. 16 *Source code visualisasi params*

Source code pada Gambar 5.16 adalah melakukan pemindaian grid (*grid search*) pada model *Support Vector Machine* (SVM) dengan berbagai kombinasi parameter C, gamma, dan jenis kernel. Melalui *nested loops*, *source code* ini menginisialisasi dan melatih model SVM pada setiap kombinasi parameter yang telah ditentukan menggunakan SVC dari library *sklearn*. Performa model dievaluasi dan disimpan dalam matriks performa. Kemudian hasilnya disajikan sebagai serangkaian *heatmap* menggunakan *seaborn* dan *matplotlib*, menampilkan hasil dari performa SVM untuk setiap jenis kernel berdasarkan variasi parameter C dan gamma.

Untuk membantu analisis visual performa model terhadap kombinasi parameter yang berbeda, berikut visualisasi performa dari pemodelan yang telah didapatkan dengan menggunakan karnel *polynomial* terhadap nilai-nilai dari parameter C, dan juga nilai-nilai dari parameter *Gamma* dengan hasil visualisasi berikut:



Gambar 5. 17 Hasil visualisasi params

Dari visualisasi hasil *tunning* sentimen analisis dengan data teks cuitan dari media sosial Twitter pada Gambar 5.17 diatas dapat dibaca untuk hasil terbaik dengan berpatokan pada akurasi tertinggi yaitu yang ditepati pada kolom ketiga untuk C dengan nilai 10 dan baris kelima untuk gamma dengan nilai 1 kemudian dengan hasil yang paling tinggi didapat 0.89. setelah mendapatkan nilai-nilai tersebut, selanjutnya adalah melakukan penggambaran untuk visualisasi karnel yang telah didapat yaitu karnel *polynomial* dengan tampilan persebaran data serta dapat melihat pemisah antar data yang didapat atau *hyperplane*.

Berikut adalah *source code* visualisasi karnel *polynomial* dari tunning yang telah dilakukan model SVM dengan mencari *hyperplane*:

```
# Reduksi dimensi ke 2D
pca = PCA(n_components=2)
X_2d = pca.fit_transform(X_train_combined)

# Latih model SVM dengan parameter terbaik dari GridSearchCV
pada data yang sudah direduksi dimensi
best_svm = SVC(C=10, gamma=0.1, kernel='poly')
best_svm.fit(X_2d, y_train)

# Visualisasi hyperplane
xx, yy = np.meshgrid(np.linspace(X_2d[:, 0].min(), X_2d[:,
0].max(), 100),
np.linspace(X_2d[:, 1].min(), X_2d[:, 1].max(), 100))

Z = best_svm.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

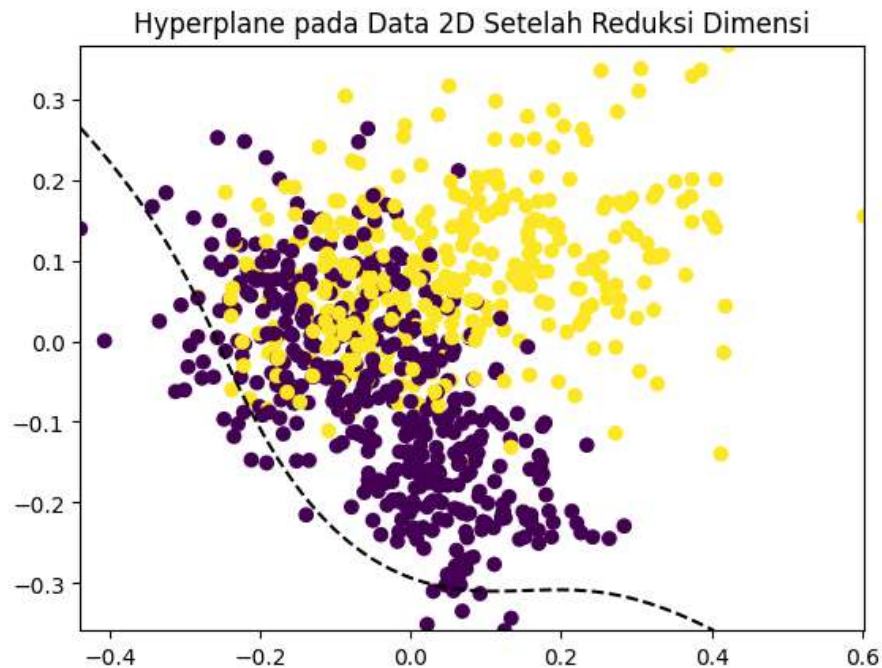
plt.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1],
linestyles=['--', '-', '--'])

plt.scatter(X_2d[:, 0], X_2d[:, 1], c=y_train, cmap='viridis')
plt.title("Hyperplane pada Data 2D Setelah Reduksi Dimensi")
plt.show()
```

Gambar 5. 18 *Source code* visuliasasi hyperplane

Source code pada Gambar 5.18 di atas diterapkan dengan menggunakan analisis PCA (*Principal Component Analysis*) untuk mereduksi dimensi fitur ke 2D, dikarenakan hasil dari *feature extraxtion* yang dihasilkan oleh *FastText* adalah berdimensi 256D. Setelah reduksi, model *Support Vector Machine* (SVM) dilatih dengan parameter terbaik dari *GridSearchCV* menggunakan data yang sudah direduksi. Selanjutnya, kode memvisualisasikan hyperplane dari model SVM pada data 2D, menampilkan garis keputusan yang dibuat oleh model untuk memisahkan kelas, dengan warna dan pola garis yang merepresentasikan batas keputusan yang dihasilkan oleh model terhadap kelas-kelas pada data yang direduksi ke 2 dimensi.

Berikut adalah hasil visualisasi dengan menggunakan kernel *polynomial* dataset yang digunakan mempunyai karakter data yang tidak terstruktur kernel ini juga cocok untuk memecahkan masalah klasifikasi pada dataset pelatihan yang dinormalisasi sesuai dengan namanya, pada klasifikasi model ini adalah dengan mencari garis pemisah antar kelas dengan hasil berikut:



Gambar 5. 19 Hasil *hyperplane*

Pada dapat dijelaskan pada Gambar 5.19 dalam pengklasifikasian dan penarikan garis *hyperplane* memiliki dua kelas yang diwakili dalam kelas ungu adalah kelas positif dan kelas kuning adalah kelas negatif yang dipisahkan oleh garis *polynomial* berwarna hitam (*hyperplane*), kemudian area antar garis putus-putus adalah *margin* yang diperoleh berdasarkan jarak terdekat antara *hyperlane* dengan kelas yang ingin dipisahkan dan setiap kelas yang berperan sebagai penentu *margin* dikenal dengan istilah *support vector*. Dalam menentukan *hyperplane* metode *Support Vector Machine* (SVM) akan memilih margin yang paling besar *maximum margin* sebagai hasil akhir dengan hasil yang cukup baik untuk perpisahan kedua data tersebut.

5.6 Validation

Pada langkah terakhir proses klasifikasi analisis sentimen ini adalah melakukan pengukuran dan evaluasi model menggunakan *confusion matrix*. *Confusion matrix* adalah sebuah tabel yang digunakan dalam pemahaman performa model klasifikasi pada *machine learning* dengan cara kerjanya menggambarkan jumlah prediksi *machine* yang benar terhadap data *real* dan jumlah prediksi *machine* yang salah terhadap data *real* yang dibuat oleh model terhadap data uji, memungkinkan untuk mengevaluasi sejauh mana model kita efektif dalam mengklasifikasikan data berdasarkan empat komponen utama yaitu TP (*True Positive*), TN (*True Negative*), FN (*False Negative*) dan FP (*False Positive*). Dari empat komponen tersebut, kita dapat menghitung berbagai metrik evaluasi model seperti akurasi, presisi, recall dan nilai F1-Score, yang memberikan pandangan yang lebih mendalam tentang kinerja model klasifikasi dalam mengatasi kasus analisis sentimen menggunakan proses *machine learning* pada studi kasus sentimen positif dan negatif. Berikut merupakan *source code* implementasi untuk *confusion matrix* beserta penjelasan hasilnya:

```

cm = confusion_matrix(y_test, prediction)

fig, ax = plt.subplots(figsize=(5,5))

ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.3)

for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(x=j, y=i, s=cm[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()

```

Gambar 5. 20 Source code confusion matrix

Source code pada Gambar 5.20 tersebut digunakan untuk menghasilkan dan menampilkan *confusion matrix* dari hasil kerja model *machine learning* terhadap data uji. *Confusion matrix* ini memberikan gambaran visual tentang seberapa baik model dapat mengklasifikasikan data menjadi kategori yang benar dan seberapa sering terjadi kesalahan prediksi, yang digunakan sebagai acuan atau untuk menentukan hasil dari perhitungan *accuracy*, *precision*, *Recall* dan *f1-score* dengan hasil *confusion matrix* sebagai berikut:

		Predicted	
		Positif	Negatif
Actual	Positif	94	15
	Negatif	13	76

Gambar 5. 21 Hasil tabel *confusion matrix*

Hasil tabel *confusion matrix* yang didapatkan pada Gambar 5.21 diatas dapat dijelaskan berdasarkan empat komponen utama yang didapatkan pada masing-masing kolom dengan penjelasan yaitu:

- TP (*True Positive*), sebanyak 94 data positif yang terklasifikasi benar.
- TN (*True Negative*), sebanyak 76 data negatif yang terklasifikasi benar.
- FP (*false positive*), yaitu sebanyak 13 data positif terklasifikasi salah.
- FN (*False Negative*), yaitu sebanyak 15 data negatif terklasifikasi salah.

Dari informasi tabel *confusin matrix* yang sudah didapat, selanjutnya adalah melakukan perhitungan *accuracy*, *preciission*, *Recall* dan *f1-score*. Untuk mengukur sejauh mana keberhasilan model dalam mengklasifikasikan data.

Accuracy (A) persentase prediksi benar dari *true positif* dan *true negative*.

$$A = \frac{(TP+TN)}{(TP+FP+FN+TN)} = \% \quad (5.11)$$

$$A = \frac{(94+76)}{(94+13+15+76)} = 0.85858585$$

$$Accuracy = 85\%$$

Precision (P) persentase prediksi benar dari dari seluruh nilai *positif*.

$$P = \frac{(TP)}{(TP+FP)} = \% \quad (5.12)$$

$$P = \frac{(94)}{(94+13)} = 0.87850467$$

$$Precision = 87\%$$

Recall (R) persentase prediksi *positif* dibandingkan dengan *true positif*.

$$R = \frac{(TP)}{(TP+FN)} = \% \quad (5.13)$$

$$R = \frac{(94)}{(94+15)} = 0.86238532$$

$$Recall = 86\%$$

F1-Score (F) merupakan perbandingan rata-rata *precision* dan *recall*.

$$F = \frac{2(P \times R)}{P+R} = \% \quad (5.14)$$

$$F = \frac{2(0.87850467 \times 0.86238532)}{0.87850467 + 0.86238532} = 0.87037037$$

$$F1 - Score = 87\%$$

BAB VI

PEMBAHASAN

Pada bagian sub bab ini membahas hasil pengujian dari kedua model yang telah dilakukan, peneliti akan fokus pada presentasi dan analisis faktor yang mempengaruhi pemrosesan hasil pengujian yang telah dilakukan terhadap kedua teknik yang dibandingkan yaitu pemrosesan *feature extraction* dengan teknik *GloVe* dan *Fasttext* untuk analisis sentimen dengan data berbentuk teks dari cuitan Twitter.

6.1 Hasil Pengujian

Untuk mengevaluasi hasil proses penerapan teknik pada *feature extraction* yang tepat agar mendapatkan hasil yang paling optimal serta akurat dari berbagai perbandingan yang diterapkan yaitu dengan melakukan pengukuran dengan menggunakan model *confusion matrix* sebagai acuan penentuan nilai *accuracy*, *precision*, *recall* dan *f1-score* yang telah dilakukan sebelumnya dengan mendapatkan perbandingan hasil kedua teknik sebagai berikut berikut:

Tabel 6. 1 Perbandingan hasil *confusion matrix*

Confusion Matrix	GloVe + SVM	FastText + SVM
TP (<i>True Positive</i>)	93	94
TN (<i>True Negative</i>)	73	76
FP (<i>false positive</i>)	16	13
FN (<i>False Negative</i>)	16	15

Dari perbandingan hasil *confusion matrix* yang telah didapatkan pada Tabel 6.1 dapat dijelaskan secara terperinci untuk perolehan hasil nilai yang didapat dari kedua teknik pada masing-masing parameter untuk menentukan dan mengetahui jenis keunggulan pada setiap teknik yang telah dilakukan pengujian. Pendekatan ini memberikan landasan yang kuat untuk memahami kinerja dan potensi setiap teknik yang di ujicobakan dengan penjelasan sebagai berikut:

- TP (*True Positive*): *Machine* dapat memprediksi bahwa sentimen tersebut positif dan memang benar dibuktikan dalam data aslinya sentimen tersebut adalah sentimen positif, dengan perolehan hasil untuk percobaan *FastText* yang lebih unggul dengan selisih satu angka yaitu 93 berbanding 94. Dengan catatan semakin besar nilai TP maka semakin akurat *machine* dalam klasifikasi.
- TN (*True Negative*): *Machine* dapat memprediksi bahwa sentimen tersebut negatif dan memang benar dibuktikan secara *real* atau dalam data aslinya sentimen tersebut adalah sentimen negatif, dengan perolehan hasil untuk percobaan *FastText* lagi yang lebih unggul dengan selisih tiga angka yaitu 73 berbanding 76. Dengan catatan semakin besar nilai TN maka semakin akurat *machine* dalam proses klasifikasi.
- FP (*False Positive*): *Machine* memprediksi bahwa sentimen tersebut positif dan ternyata prediksi salah sentimen tersebut adalah sentimen negatif, dengan perolehan hasil untuk percobaan *FastText* lagi yang lebih unggul dengan selisih tiga angka yaitu 13 berbanding 16. Dengan catatan semakin kecil nilai FP maka semakin sedikit *machine* dalam melakukan kesalahan untuk hasil klasifikasi. Dalam keadaan ini disebut dengan kesalahan (*Type Error 1*).
- FN (*False Negative*): *Machine* memprediksi bahwa sentimen tersebut negatif dan ternyata prediksi salah, sentimen tersebut adalah sentimen positif, dengan perolehan hasil untuk percobaan teknik *FastText* lagi yang lebih unggul dibandingkan dengan teknik *GloVe* dengan selisih hanya satu angka yaitu *FastText*+SVM 15 berbanding *GloVe*+SVM 16. Dengan catatan semakin kecil nilai FN maka semakin sedikit *machine* dalam melakukan kesalahan klasifikasi. Dalam keadaan ini disebut dengan kesalahan (*Type Error 2*).

Seperti yang telah dijelaskan dengan detail terkait hasil dari *confusion matrix*, dari beberapa hasil yang telah dipaparkan bahwa FN merupakan kesalahan tipe 2 (*Type Error 2*) dimana kesalahan ini sangat berbahaya dibandingkan dengan kesalahan tipe 1 (*Type Error 1*) pada kasus ini karena jika *machine* melakukan klasifikasi teks untuk sentimen analisis memberikan label negatif dan pada nyatanya kalimat tersebut adalah kalimat positif maka akan timbul kesalahan yang fatal dengan melalaikan tujuan utama yaitu *menjadikan* suatu program yang berguna bagi masalah serta untuk menghindari perilaku atau kejadian prasangka buruk pada hasil penentuan label keputusan pada setiap komentar. Dengan hasil penerapan teknik *FastText* dengan SVM menjadi model yang memiliki nilai terbaik untuk hasil *confusion matrix* dalam studi kasus analisis sentiment dari beberapa instrument yang ada pada table *confusion matrix* dan juga pada teknik *FastText* dengan SVM memiliki nilai kesalahan *error tipe 2* yang kecil dibandingkan dengan hasil teknik *GloVe* dengan algoritma *Support Vector Machine* (SVM).

Dari hasil dan kesimpulan *confusion matrix* yang telah didapatkan, selanjutnya nilai-nilai tersebut digunakan untuk menghitung berapakah *performance matrix* guna untuk mengetahui kinerja model yang telah dibuat. Pada bagian ini dapat dipahami beberapa *performance matrix* yang digunakan yaitu *accuracy*, *precision*, *recall* dan *f1-score* yang telah dilakukan sebelumnya dengan mendapatkan perbandingan hasil kedua teknik sebagai berikut berikut:

Tabel 6. 2 Perbandingan hasil *performance matrix*

Parameter	GloVe + SVM	FastText + SVM
<i>Accuracy</i>	83%	85%
<i>precision</i>	85%	87%
<i>Recall</i>	85%	86%
<i>f1-score</i>	85%	87%

Dari perbandingan hasil *performance matrix* yang telah didapatkan pada Tabel 6.2 dapat dijelaskan secara terperinci untuk perolehan nilai dari kedua metode pada masing-masing parameter untuk menentukan dan mengetahui jenis keunggulan pada setiap teknik yang telah dilakukan pengujian. Pendekatan ini merupakan hasil akhir yang memberikan hasil kinerja *machine* secara menyeluruh:

- *Accuracy* menggambarkan seberapa akurat model dapat mengklasifikasikan dengan benar. Maka *accuracy* merupakan rasio prediksi benar (sentimen positif dan sentimen negatif) dengan keseluruhan data. Dengan kata lain, *accuracy* merupakan tingkat kedekatan nilai prediksi dengan nilai aktual (sebenarnya). Nilai *accuracy* dapat diperoleh dengan hasil yang lebih tinggi pada teknik *FastText* dengan perbedaan dua angka dari teknik *GloVe* yakni 83% berbanding dengan 85%.
- *Precision* menggambarkan tingkat keakuratan antara data yang diminta dengan hasil prediksi yang diberikan oleh model. Maka, *precision* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif. Nilai *precision* dapat diperoleh masih dengan hasil yang lebih tinggi pada teknik *FastText* dengan perbedaan dua angka dari teknik *GloVe* yakni 85% berbanding dengan 87%.
- *Recall* menggambarkan seberapa berhasilkah model dalam menemukan kembali sebuah informasi yang ada pada klasifikasi yang telah dilakukan pemrosesan untuk analisis sentimen. Maka pada dasarnya *recall* merupakan rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Nilai *recall* dapat hasil yang lebih tinggi lagi pada teknik *FastText* dengan perbedaan tipis hanya satu angka yakni 85% berbanding dengan 86%.

- *F1-score* merupakan kombinasi dari *precision* dan *recall* yang memberikan gambaran keseluruhan tentang kinerja model. Nilai *f1-score* yang tinggi menunjukkan keseimbangan yang baik antara ketepatan dan keberhasilan model dalam menemukan informasi. Dalam kasus ini *f1-score* juga masih menunjukkan hasil yang lebih tinggi pada teknik *FastText*, dengan perbedaan dua angka dari teknik *GloVe* yakni 85% berbanding 87%. *F1-score* berguna untuk mengevaluasi performa model secara keseluruhan, terutama ketika keseimbangan antara *precision* dan *recall* sangat penting.

Berdasarkan hasil paparan pengujian performa model pada analisis sentimen dengan data teks berasal dari cuitan Twitter menggunakan teknik *GloVe* dan SVM dibandingkan dengan *FastText* dan SVM, dengan semua perbandingan hasil yang telah didapatkan dan dibahas sebelumnya dapat disimpulkan bahwa teknik *FastText* yang disandingkan dengan metode SVM menunjukkan kinerja yang lebih unggul dari semua teknik yang digunakan dalam penilaian yaitu *confusion matrix* dan *performance matrix*. Meskipun perbedaan hasil nilai antara kedua teknik tersebut tidak terlalu signifikan namun nilai *accuracy*, *precision*, *recall* dan *f1-score* pada *FastText* *consistently* selalu lebih tinggi dibandingkan *GloVe* dengan metode SVM. Hal ini menandakan bahwa kombinasi antara metode vektorisasi teks dengan *FastText* dan model SVM mampu memberikan hasil klasifikasi yang lebih akurat dan seimbang dalam mengenali sentimen positif dan negatif pada data dalam bentuk teks yang bersumber dari media social cuitan *Twitter*. Oleh karena itu, untuk tugas analisis sentimen pada studi kasus ini teknik *FastText* dan SVM dapat dianggap sebagai pilihan yang lebih optimal dan unggul untuk mendapatkan hasil yang lebih baik serta akurat dibandingkan dengan penggunaan model *GloVe* dengan SVM.

6.2 Hasil Evaluasi

Beberapa faktor yang dapat mempengaruhi kinerja dari penerapan *feature extraction* pada proses analisis sentimen, seperti yang di lakukan dengan teknik *GloVe* dan *FastText*, melibatkan pemilihan metode *feature extraction*, sumber data dan karakteristik dataset. Berikut beberapa faktor utama:

- Metode *Feature Extraction*:

Kemampuan representasi teks, metode *feature extraction* harus dapat menggambarkan teks dengan baik, menangkap makna dan relasi antar kata. *GloVe* dan *FastText* memiliki pendekatan yang berbeda dalam merepresentasikan kata-kata (*word embeddings* vs *subword embeddings*), dan performanya dapat dipengaruhi oleh sifat dataset.

- Ukuran dan Kualitas Dataset:

Ukuran dataset, jumlah data yang cukup besar dapat membantu model untuk memahami variasi dan kompleksitas dalam bahasa. Dataset yang kecil dapat menyebabkan *overfitting*. Kualitas labeling, keakuratan label pada dataset sangat penting dan kesalahan dalam labeling menyebabkan model menghasilkan prediksi yang tidak akurat.

- Tuning Parameter:

Parameter algoritma, pada metode SVM, tuning parameter C dapat mempengaruhi kinerja model serta pengaturan parameter yang tepat dapat meningkatkan kemampuan model untuk menangkap pola-pola dalam data.

- Preprocessing Teks:

Pembersihan teks, proses pembersihan teks, seperti penghapusan tanda baca atau *stemming*, dapat memengaruhi representasi kata-kata dan kinerja model.

- Distribusi Sentimen dalam Dataset:

Keseimbangan sentiment, jika dataset memiliki keseimbangan yang tidak merata antara sentimen positif dan negatif, model dapat cenderung memihak pada kelas mayoritas, menghasilkan evaluasi yang bias, ketidak seimbangannya data tersebut dapat ditangani dengan teknik *oversampling* atau *undersampling*.

- Karakteristik Bahasa dan Domain:

Spesifik domain, karakteristik teks dalam domain tertentu (misalnya, bahasa informal *Twitter*) dapat memerlukan penyesuaian khusus dalam *feature extraction* untuk meningkatkan kinerja.

Pada proses evaluasi akhir adalah dengan menentukan dan menyimpulkan model yang terbaik yang sudah dilakukan dari perbandingan beserta dengan pengujian dan evaluasi hasil, didapatkan paduan antara teknik *feature extraction* menggunakan *FastText* dan dipadukan dengan algoritma *Support Vector Machine* (SVM) memiliki hasil dan performa yang unggul dalam segala aspek pengukuran. Dimana salah satu keunggulan *FastText* juga terletak pada kemampuannya dalam menangani teks dengan variasi kata dan frasa yang lebih luas, termasuk kata-kata yang tidak umum atau bahasa *slang* yang sering muncul di platform media sosial *Twitter*. Penggunaan *FastText* juga dapat meningkatkan efisiensi komputasional karena model tersebut memperhitungkan *subword information*, memungkinkan pemodelan representasi kata yang lebih akurat. Namun, penting untuk terus mengkaji dan membandingkan kinerja model dengan dataset yang berbeda untuk memastikan generalitas hasil. Kesimpulannya akhirnya adalah, pemodelan dengan teknik *FastText* dan algoritma *Support Vector Machine* (SVM) memberikan solusi yang handal dalam analisis sentimen data teks dengan potensi untuk peningkatan kinerja dan generalitas.

BAB VII

PENUTUP

7.1 Kesimpulan

Berdasarkan hasil analisis semua percobaan yang telah dilakukan dapat disimpulkan bahwa model pada teknik *FastText* dan SVM menunjukkan kinerja yang lebih unggul. Secara khusus, model *FastText* memiliki tingkat akurasi sebesar 85%, sedangkan model *GloVe* hanya mencapai 83%. Lebih lanjut, *precision* pada model *FastText* mencapai 87%, sedangkan pada model *GloVe* hanya sebesar 85%. Meskipun *recall* model *FastText* hanya sedikit lebih tinggi daripada *GloVe* yaitu 85%: 86%, namun nilai *F1-score* yang mencapai 87% pada model *FastText* menunjukkan keseimbangan yang baik antara ketepatan dan keberhasilan dalam menemukan informasi, sedangkan model *GloVe* hanya mencapai 85%.

Kemudian dari hasil evaluasi penerapan kedua model yang telah dilakukan terdapat beberapa permasalahan yang ditemukan dalam penerapannya anatara lain. Pertama, ukuran korpus teks yang besar dapat memperlambat waktu *processing* karena memerlukan sumber daya komputasi yang signifikan untuk melatih model, penanganannya dengan melibatkan penggunaan *subset* data atau penyesuaian parameter untuk mempercepat proses pelatihan. Kedua, masalah ketidak seimbangan kelas dalam data sentimen dapat menyebabkan performa yang tidak optimal atau *machine* akan cenderung mengarah pada jumlah kelas data yang tinggi, penanganannya dengan melibatkan teknik *oversampling* dan *undersampling* untuk menyeimbangkan jumlah label kelas. Ketiga, metode SVM dapat sensitif terhadap skala fitur dan parameter yang detail, sehingga perlu dilakukan normalisasi atau proses *tunning* data untuk mendapatkan hasil optimal.

7.2 Saran

Terdapat beberapa saran yang dapat berikan oleh peneliti untuk pengembangan penelitian berikutnya, saran-saran ini bertujuan untuk memperdalam dan meningkatkan efektivitas analisis sentimen berbasis *feature extraction*, guna menangani beberapa tantangan yang mungkin muncul dalam implementasi dengan studi kasus analisis sentimen pada data teks Twitter kedepannya:

1. Penggunaan *FastText* mungkin terbatas oleh sumber daya komputasi ketika diterapkan pada korpus teks yang besar. Penelitian berikutnya untuk dapat memfokuskan pada strategi optimasi untuk melibatkan *FastText* pada korpus teks yang luas. Misalnya, peneliti dapat menjelajahi teknik pengurangan dimensi atau pendekatan *transfer learning* untuk mempercepat proses pelatihan tanpa mengorbankan kinerja model sebelumnya.
2. Penelitian mendatang dapat mengeksplorasi lebih lanjut metode untuk menangani ketidak seimbangan kelas secara efektif. Seperti menambahkan teknik *oversampling* dan *undersampling* dengan metode baru yang muncul atau mengkombinasikan teknik tersebut.

DAFTAR PUSTAKA

- Arsi, P., & Waluyo, R. (2021). Analisis Sentimen Wacana Pemindahan Ibu Kota Indonesia Menggunakan Algoritma Support Vector Machine (SVM). *Jurnal Teknologi Informasi Dan Ilmu Komputer (JTIK)*, 8(1), 147–156. <https://doi.org/10.25126/jtiik.202183944>
- Cindo, M., & Rini, D. P. (2019). Metode Klasifikasi Pada Sentimen Analisis. *Seminar Nasional Teknologi Komputer & Sains (SAINTEKS)*, 1(1), 66–70. <https://seminar-id.com/semnas-sainteks2019.html>
- Dikih Arif Wibowo, B. (2020). Analisis Sentimen Tweet Berbahasa Indonesia Tentang Vaksin Covid-19 Menggunakan Fasttext Embedding Dan Support Vector Machine. *Universitas Gadjah Mada*. <http://etd.repository.ugm.ac.id/>
- Duei Putri, D., Nama, G. F., & Sulistiono, W. E. (2022). Analisis Sentimen Kinerja Dewan Perwakilan Rakyat (DPR) Pada Twitter Menggunakan Metode Naive Bayes Classifier. *Jurnal Informatika Dan Teknik Elektro Terapan*, 10(1). <https://doi.org/10.23960/jitet.v10i1.2262>
- Faiq, M., Putro, A., & Setiawan, E. B. (2022). Analisis Sentimen Terhadap Kebijakan Pemerintah dengan Feature Expansion Metode GloVe pada Media sosial Twitter. *E-Proceeding of Engineering*, 9(1), 54–66.
- Fluorida Fibrianda, M., & Bhawiyuga, A. (2018). Analisis Perbandingan Akurasi Deteksi Serangan Pada Jaringan Komputer Dengan Metode Naïve Bayes Dan Support Vector Machine (SVM). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, 2(9), 3112–3123. <http://j-ptiik.ub.ac.id>
- Fransiska, S., & Irham Gufroni, A. (2020). Sentiment Analysis Provider by.U on Google Play Store Reviews with TF-IDF and Support Vector Machine (SVM) Method. *Scientific Journal of Informatics*, 7(2), 2407–7658. <http://journal.unnes.ac.id/nju/index.php/sji>
- Giovani, A. P., Ardiansyah, A., Haryanti, T., Kurniawati, L., & Gata, W. (2020). Analisis Sentimen Aplikasi Ruang Guru Di Twitter Menggunakan Algoritma Klasifikasi. *Jurnal Teknoinfo*, 14(2), 115. <https://doi.org/10.33365/jti.v14i2.679>
- Hikmawan, S., Pardamean, A., Nur Khasanah, S., Mandiri, N., Damai No, J., Jati Barat, W., & Selatan, J. (2020). Sentimen Analisis Publik Terhadap Joko Widodo Terhadap Wabah Covid 19 Menggunakan Metode Machine Learning. *Jurnal Kajian Ilmiah (JKI)*, 20(2), 167–176. <http://ejournal.ubharajaya.ac.id/index.php/JKI>
- Imron, A. (2019). Analisis Sentimen Terhadap Tempat Wisata Di Kabupaten Rembang Menggunakan Metode Naive Bayes Classifier. 1–45.
- Indriani, A. (2020). Analisa Perbandingan Metode Naïve Bayes Classifier Dan K-Nearest Neighbor Terhadap Klasifikasi Data. *Sebatik*.
- Jagdale, R. S., Shirsat, V. S., & Deshmukh, S. N. (2019). Sentiment analysis on product reviews using machine learning techniques. *Advances in Intelligent Systems and Computing*, 768, 639–647. https://doi.org/10.1007/978-981-13-0617-4_61

- Maryanto, B. (2017). Big Data Dan Pemanfaatannya Dalam Berbagai Sektor. *Media Informatika*, 16(2).
- Muara Sains, J., Ilmu Kesehatan, dan, Trisari Harsanti Putri, W., & Hendrowati, R. (2018). Penggalan Teks Dengan Model Bag Of Words Terhadap Data Twitter. *Jurnal Muara Sains, Teknologi, Kedokteran, Dan Ilmu Kesehatan*, 2(1), 129.
- 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.
- Octaviani, A., & Dewi, P. (2020). Big Data di Perpustakaan dengan Memanfaatkan Data Mining. *ANUVA*, 4(2), 223–230.
- Permatasari, P. A., Linawati, L., & Jasa, L. (2021). Survei Tentang Analisis Sentimen Pada Media Sosial. *Majalah Ilmiah Teknologi Elektro*, 20(2), 177. <https://doi.org/10.24843/mite.2021.v20i02.p01>
- Putri, I. E., Rahmawati, D., & Yufis Azhar, ; (2020). Comparison Of Data Mining Classification Methods To Detect Heart DiseaSE. *Jurnal PILAR Nusa Mandiri*, 16(2), 213–218. <https://doi.org/10.33480/pilar.v16i2.1481>
- Rohanah, A., Rianti, D. L., Sari, B. N., Informatika, T., & Karawang, U. S. (2021). Perbandingan Naïve Bayes Dan Support Vector Machine Untuk Klasifikasi Ulasan Pelanggan Indihome. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)*, 6(1), 23–30.
- Sabrila, T. S., Sari, V. R., & Minarno, A. E. (2021). Analisis Sentimen Pada Tweet Tentang Penanganan Covid-19 Menggunakan Word Embedding Pada Algoritma Support Vector Machine Dan K-Nearest Neighbor. *Fountain of Informatics Journal*, 6(2), 69. <https://doi.org/10.21111/fij.v6i2.5536>
- Sari, F. V., & Wibowo, A. (2019). Analisis Sentimen Pelanggan Toko Online Jd.Id Menggunakan Metode Naïve Bayes Classifier Berbasis Konversi Ikon Emosi. *Jurnal SIMETRIS*, 10(2).
- Wahyono, T. (2018). *Fundamental of Python for Machine Learning: Dasar-Dasar Pemrograman Python untuk Machine Learning dan Kecerdasan Buatan* (Vol. 1). <https://www.researchgate.net/publication/330441937>
- Wahyu Kurniawan, F., & Maharani, W. (2020). Analisis Sentimen Twitter Bahasa Indonesia dengan Word2Vec. *E-Proceeding of Engineering*, 7(2), 7821–7828. <https://code.google.com>
- Wibawa, A. P., Guntur, M., Purnama, A., Fathony Akbar, M., & Dwiyanto, F. A. (2018). Metode-metode Klasifikasi. *Prosiding Seminar Ilmu Komputer Dan Teknologi Informasi*, 3(1).