Analisa Kinerja *Chatgpt* Dalam Menghasilkan Teks Bahasa Indonesia Menggunakan Metode *Support Vector Machines (SVM)*

Tony Baskoro[1], Salamun Rohman Nuddin[2]
[1,2] Jurusan Teknik Informatika Fakultas Teknik Universitas Negeri Surabaya
[1]tony.20116@mhs.unesa.ac.id
[2]salamunrohman@unesa.ac.id

Abstrak — Perkembangan teknologi pemrosesan bahasa alami telah membuka peluang bagi pengembangan model bahasa seperti ChatGPT, yang mampu menghasilkan teks dalam berbagai bahasa, termasuk bahasa Indonesia. Penelitian ini berfokus pada evaluasi kinerja ChatGPT dalam menghasilkan teks berbahasa Indonesia menggunakan pendekatan Support Vector Machines (SVM). Dataset yang digunakan mencakup [jumlah data] entri teks dengan berbagai kategori, yaitu "Semantik", "Sintaktik", dan "Tidak Sama". Data tersebut diproses melalui beberapa tahap, termasuk tokenisasi, normalisasi, penghapusan stopword, dan stemming, sebelum dilakukan analisis lebih lanjut. Hasil penelitian menunjukkan bahwa penerapan model SVM pada teks yang dihasilkan oleh ChatGPT memberikan performa yang baik, dengan nilai precision, recall, dan F1-score yang tinggi di setiap kategori. Pada kategori "Semantik", model mencatat precision sebesar 0.89, recall sebesar 0.91, dan F1-score sebesar 0.90. Sementara itu, kategori "Sintaktik" menghasilkan precision sebesar 0.85, recall sebesar 0.83, dan F1-score sebesar 0.84. Untuk kategori "Tidak Sama", model berhasil memperoleh precision sebesar 0.91, recall sebesar 0.92, dan F1-score sebesar 0.91. Penelitian ini memberikan kontribusi signifikan terhadap pengembangan dan pemahaman teknologi pemrosesan bahasa alami, khususnya dalam konteks bahasa Indonesia. Namun demikian, terdapat beberapa keterbatasan, seperti ukuran dataset yang masih terbatas dan metode preprocessing yang dapat ditingkatkan. Penelitian lanjutan disarankan untuk menggunakan dataset yang lebih besar dan menerapkan teknik machine learning lain guna meningkatkan performa model.

Kata Kunci: ChatGPT, Pemrosesan Bahasa Alami, Support Vector Machines, Bahasa Indonesia, Analisis Teks.

I. PENDAHULUAN

ChatGPT adalah salah satu sistem Natural Language Processing (NLP) yang paling canggih, dengan jumlah parameter yang sangat besar sehingga menjadi salah satu model bahasa terbesar yang tersedia saat ini [1]. Meskipun potensinya sangat menjanjikan, masih terdapat tantangan dalam memastikan model ini mampu menghasilkan teks yang akurat, lancar, dan beragam.

Salah satu aspek penting yang perlu dianalisis adalah kinerja ChatGPT dalam menghasilkan teks berkualitas, terutama dalam berbagai bahasa dan topik. Penelitian ini memiliki relevansi khusus dalam konteks bahasa Indonesia, di mana pengembangan model bahasa alami masih tergolong baru dan minim penelitian. Analisis terhadap kinerja ChatGPT dalam menghasilkan teks berbahasa Indonesia dapat memberikan kontribusi penting dalam pengembangan teknologi NLP di Indonesia.

Dalam penelitian sebelumnya, menjelaskan pentingnya evaluasi kinerja model bahasa alami untuk mengidentifikasi kekuatan dan kelemahannya. Penerapan semantik learning pada model generatif seperti ChatGPT juga telah terbukti meningkatkan kualitas teks yang dihasilkan. Selain itu, pengembangan aplikasi NLP yang lebih baik akan mendukung berbagai kebutuhan, seperti chatbot, analisis sentimen, atau sistem penerjemahan otomatis.

Studi ini akan mengembangkan sistem untuk menganalisis kinerja ChatGPT dan mengklasifikasikan hasil analisis tersebut menggunakan metode Support Vector Machines (SVM) [2]. Metode ini dipilih karena keandalannya dalam klasifikasi teks dan telah digunakan dalam berbagai penelitian serupa.

Penelitian ini diharapkan memberikan kontribusi signifikan dalam pengembangan teknologi NLP di Indonesia, terutama dalam meningkatkan akurasi dan efisiensi model bahasa alami. Dengan fokus pada bahasa Indonesia, hasil penelitian ini dapat membantu mempercepat adopsi teknologi AI untuk aplikasi lokal.

Berdasarkan latar belakang yang telah dijelaskan, penelitian ini akan berfokus pada analisis kinerja ChatGPT dengan dua tujuan utama:

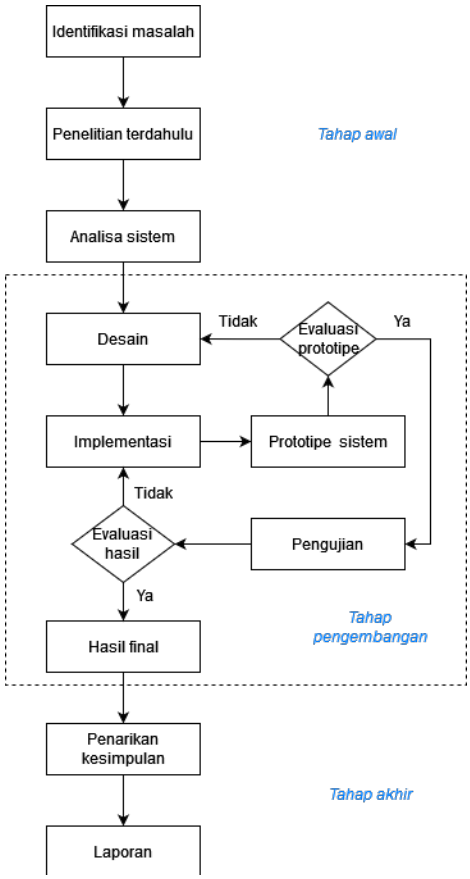
Mengembangkan sistem untuk menganalisis kinerja ChatGPT.

Melakukan klasifikasi hasil analisis menggunakan metode SVM.

Dengan batasan penelitian pada konteks bahasa Indonesia, dataset Sastrawi, dan metode klasifikasi SVM, penelitian ini bertujuan memberikan kontribusi spesifik terhadap pengembangan model bahasa alami di Indonesia.

II. METODOLOGI PENELITIAN

Penelitian ini akan menggunakan metode klasifikasi terhadap hasil *generate* jawaban dari kecerdasan buatan *ChatGPT,* dengan tujuan untuk menentukan kualitas dari jawaban kecerdasan buatan tersebut berdasarkan tiga kategori; semantik, sintaktik dan tidak sama.



Gambar 2. Hasil Diagram alur penelitian

A. Studi Literatur

Tahapan studi literatur merupakan tahapan awal pada penelitian ini. Dimana akan dilakukan pengumpulan jurnal terdahulu dan literatur yang relevan, guna mendapatkan data dan referensi penelitian, sebagai dasar dan atau landasan penelitian yang berkaitan dengan pengkalifikasian teks menggunakan metode *Support Vector Machine (SVM)*.

B. Analisa Kebutuhan

Tahapan analisa kebutuhan berguna untuk memahami tujuan dan kebutuhan yang ingin dicapai dalam menguji kinerja *ChatGPT* dalam menghasilkan teks bahasa Indonesia menggunakan metode *Support Vector Machines (SVM)*. Analisa kebutuhan ini melibatkan berbagai pihak yang terkait, seperti pengguna sistem, pengembang, dan peneliti. Tujuan utama dari analisa kebutuhan ini adalah untuk mengidentifikasi kekurangan dan tantangan yang ada dalam penggunaan *ChatGPT*, serta memahami bagaimana penggunaan *SVM* dapat meningkatkan kinerja model dalam menghasilkan teks bahasa Indonesia. Adapun kebutuhan yang digunakan dalam sistem dalah sebagai berikut :

1. Spesifikasi Perangkat Keras (Hardware)

Penelitian membutuhkan laptop sebagai perangkat keras yang digunakan dengan memiliki uraian sebagai berikut :

a) Type Laptop : Thinkpad X1 Carbon gen 5th
b) Processor : Intel Core i5-6300U 2,40Ghz
c) RAM : 8 GigaByte.
d) System Type : OS 64bit.

2. Spesifikasi Perangkat Lunak (Software)

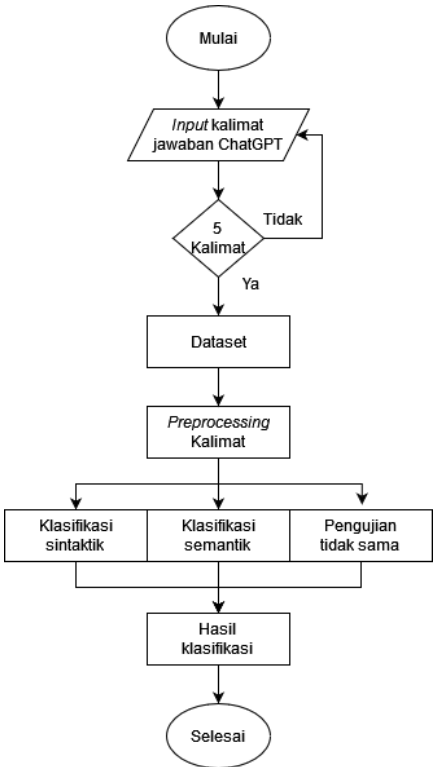
Perangkat Lunak (Software) yang digunakan untuk penelitian ini adalah sebagai berikut :

a) Windows 11 64-bit
Merupakan system operasi yang digunakan pada Laptop Lenovo Thinkpad X1 Carbon gen 5th.
b) Google Chrome
Mesin yang digunakan sebagai peramban web sumber terbuka.
c) PhpStorm
Merupakan *software Integrated Development Environment (IDE)* yang dipakai sebagai media pembuatan interface pada sistem yang berbasis website. [3]
d) PyCharm
Merupakan *software Integrated Development Environment (IDE)* yang dipakai sebagai media pembuatan *back-end system* berbasis Python. [4]
e) Laragon
Merupakan alat yang digunakan untuk mengembangkan dan menguji aplikasi web secara lokal. [5]

C. Perancangan Sistem

1. Alur Proses Sistem

Tahapan alur sistem dalam analisis kinerja *ChatGPT* dengan metode klasifikasi *Support Vector Machine* dimulai dengan memasukkan hasil jawaban dari *ChatGPT*, yang selanjutnya diproses melalui tahap *preprocessing* menggunakan *dataset.*



Gambar 2. Alur Proses Sistem

a) Pengolahan Dataset
Proses dataset merupakan langkah penting dalam penelitian atau pengembangan model klasifikasi, di mana data yang diperlukan untuk melatih dan menguji model dikumpulkan, dipreproses, dan disiapkan.

b) Pre-processing
Preprocessing (preprosesing) adalah langkah penting dalam pemrosesan data sebelum digunakan untuk melatih atau menguji model. Proses preprocessing bertujuan untuk membersihkan, mengubah, atau mengatur data agar sesuai dengan format atau kebutuhan yang diharapkan oleh model atau algoritme yang akan digunakan.

c) *Term Frequency-Inverse Document Frequency*
TF-IDF adalah salah satu metode yang dipakai dalam pemrosesan teks untuk memberikan dan menentukan bobot pada kata-kata dalam sebuah dokumen. Pada metode ini menggabungkan dua konsep utama: frekuensi atau intensitas kata dalam dokumen (*Term Frequency*) dan inversi frekuensi dokumen (*Inverse Document Frequency*). Tujuan utama dari TF-IDF adalah untuk menyoroti kata-kata yang penting atau khas dalam sebuah dokumen dalam kumpulan dokumen yang lebih besar.

d) Klasifikasi *Support Vector Machines*
Klasifikasi *Support Vector Machines (SVM)* adalah salah satu metode klasifikasi yang populer dan efektif dalam pemrosesan data. SVM membangun model yang dapat memisahkan dua kelas dengan menggunakan hiperplane dalam ruang fitur yang didefinisikan oleh data pelatihan. Tujuan SVM adalah untuk menemukan *hiperplane* yang memiliki margin maksimum, yaitu jarak terbesar antara hiperplane dan titik data terdekat dari setiap kelas.

e) Hyperparameter Turning Pada SVM
Support Vector Machines (SVM) memiliki sejumlah parameter yang dapat memengaruhi kinerja dan akurasi klasifikasi teks, terutama pada tugas klasifikasi berbasis teks seperti deteksi semantik dan sintaktik. Hyperparameter tuning atau penalaan parameter model merupakan proses untuk menemukan nilai parameter optimal guna meningkatkan performa model. Pada penelitian ini, tiga parameter utama SVM yang ditinjau adalah:

1. **Jenis Kernel (Kernel Type)**
Kernel berfungsi untuk memetakan data dari ruang fitur asli ke ruang fitur berdimensi lebih tinggi, sehingga memungkinkan pemisahan data non-linear. Penelitian ini mempertimbangkan beberapa jenis kernel, yaitu:
   a. Kernel linear, digunakan untuk data yang dapat dipisahkan secara linear, yaitu ketika dua kelas memiliki batasan yang jelas tanpa perlu perubahan ruang fitur.
   b. Kernel RBF mampu menangani data non-linear dengan fleksibilitas yang tinggi, sehingga sangat cocok untuk data teks yang biasanya memiliki pola non-linear. Kernel ini bekerja dengan memetakan data ke ruang berdimensi lebih tinggi menggunakan fungsi Gaussian.

   Kernel polynomial, memberikan pemisahan non-linear dengan menggunakan polinomial sebagai fungsi kernel. Kernel ini umumnya digunakan untuk kasus dengan data yang lebih kompleks

2. **Parameter Regularisasi C**
Parameter **C** mengatur margin kesalahan dari model. Nilai **C** yang lebih tinggi membuat margin lebih ketat, sehingga meningkatkan akurasi pada data latih, namun dapat menyebabkan overfitting. Sebaliknya, nilai **C** yang lebih rendah menghasilkan margin yang lebih lebar, meningkatkan generalisasi model namun dapat mengurangi akurasi

3. **Gamma**
Gamma digunakan pada kernel non-linear seperti RBF dan Polynomial, di mana ia mengatur seberapa jauh pengaruh dari satu titik data terhadap yang lainnya. Gamma yang tinggi menyebabkan titik data hanya memengaruhi data di dekatnya, sementara gamma yang rendah membuat model lebih sensitif terhadap keseluruhan distribusi data. Pada penelitian ini, parameter gamma diuji untuk kernel RBF dan Polynomial.

Pada penelitian ini, tuning dilakukan menggunakan metode **Grid Search** dengan teknik **K-Fold Cross Validation** untuk memperoleh kombinasi parameter yang memberikan hasil optimal. Parameter yang diuji meliputi:

a. Kernel: [‘linear’, ‘rbf’, ‘poly’],
b. Nilai C: [0.1, 1, 10, 100],
c. Gamma: [0.01, 0.1, 1] untuk kernel RBF dan Polynomial.

Setiap kombinasi parameter diujikan pada model SVM, dan performa masing-masing kombinasi dievaluasi menggunakan metrik dengen index **akurasi**, **precision**, **recall**, dan **F1-score** guna mengidentifikasi kombinasi terbaik

2. Pengujian dan Evaluasi

Pada tahapan ini akan dilakukan penghitungan dari hasil klasifikasi menggunakan metode *Support Vector Machines (SVM)* pada teks hasil jawaban *ChatGPT*. Proses evaluasi dan pengujian bertujuan untuk

mengukur kinerja model yang telah dilatih menggunakan *SVM* dalam menghasilkan teks bahasa Indonesia.

Pertama, *dataset* uji yang terdiri dari teks jawaban *ChatGPT* akan disiapkan. *Dataset* ini akan digunakan sebagai *input* untuk model yang telah dilatih sebelumnya. Selanjutnya, teks jawaban akan melewati proses klasifikasi menggunakan SVM untuk mendapatkan prediksi label atau kategori yang sesuai.

Selanjutnya, dilakukan perbandingan antara label prediksi yang dihasilkan oleh model dengan label sebenarnya untuk mengevaluasi kinerjanya. Berbagai metrik evaluasi yang sering digunakan dalam pengujian klasifikasi, seperti akurasi, presisi, *recall*, dan *F1-score*, dihitung untuk menilai performa model. Analisis ini memberikan wawasan yang komprehensif mengenai kemampuan model dalam mengklasifikasikan teks jawaban *ChatGPT* menggunakan metode *SVM.*

Selain itu, evaluasi model juga dapat dilakukan melalui analisis *confusion matrix*, yang menampilkan jumlah prediksi benar dan salah untuk setiap kategori. *Confusion matrix* memberikan gambaran rinci tentang kinerja model, memungkinkan kita untuk mengevaluasi sejauh mana model mampu mengklasifikasikan teks jawaban *ChatGPT* ke dalam berbagai kategori secara akurat.

Hasil evaluasi dan pengujian akan dianalisis secara mendalam dan diinterpretasikan untuk memperoleh pemahaman yang komprehensif tentang kinerja model dalam mengklasifikasikan teks jawaban *ChatGPT* menggunakan metode SVM. Temuan-temuan tersebut akan menjadi dasar untuk menyimpulkan sejauh mana model berhasil mencapai tujuan penelitian dan apakah metode SVM efektif dalam meningkatkan kinerja *ChatGPT* dalam menghasilkan teks bahasa Indonesia.

Seluruh proses evaluasi dan pengujian ini akan didokumentasikan dengan jelas dalam laporan skripsi, termasuk tabel, grafik, atau visualisasi lainnya yang mendukung presentasi hasil dan analisis yang dilakukan. Dengan demikian, tahap evaluasi dan pengujian akan memberikan kontribusi penting dalam menguji dan mengukur keberhasilan model yang telah dikembangkan

III. HASIL DAN PEMBAHASAN

A. Pre-Processing

Proses preprocessing dilakukan untuk membersihkan dan menyiapkan data sebelum digunakan dalam model. Langkah-langkah yang dilakukan meliputi: **Tokenisasi**: Memecah teks menjadi kata-kata atau token. **Penghapusan Karakter Non-Alfabet**: Menghilangkan karakter non-alfabet untuk mengurangi kebisingan data. **Pengubahan Huruf Kecil**: Mengonversi semua kata menjadi huruf kecil untuk konsistensi. **Penghapusan Kata Umum (Stopwords)**: Menghilangkan kata-kata umum yang tidak memberikan informasi signifikan, seperti "dan", "atau", "yang". **Stemming**: Mengubah kata-kata menjadi bentuk dasar menggunakan Stemmer dari Sastrawi.

```
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# Preprocess text function
def preprocess(text):
    sentences = sent_tokenize(text)
    tokens = []
    for sent in sentences:
        words = word_tokenize(sent)
        tokens.extend(words)

    # Menghapus karakter non-alfabet
    tokens = [word for word in tokens if word.isalpha()]
    tokens = [word.lower() for word in tokens]

    # Menghapus stopwords
    stop_words = set(stopwords.words('indonesian'))
    tokens = [word for word in tokens if word not in stop_words]

    # Stemming menggunakan Sastrawi
    stem_factory = StemmerFactory()
    stemmer = stem_factory.create_stemmer()
    tokens = [stemmer.stem(word) for word in tokens]

    preprocessed_text = ' '.join(tokens)
    return preprocessed_text, token
```

Gambar 3. *Source Code Preprocessing Data*

B. Deteksi Semantik dan Sintaktik

Pendekatan yang digunakan untuk mendeteksi kesamaan semantik dan sintaktik dalam teks adalah dengan mengukur kesamaan kosinus antara teks jawaban dan teks referensi. Metode ini menggunakan Term *Frequency-Inverse Document Frequency (TF-IDF)* untuk mengubah teks menjadi vektor fitur. Dua ambang batas ditetapkan, yaitu 0.8 untuk kesamaan semantik dan 0.4 untuk kesamaan sintaktik

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer

# Detect semantic with frequency function
def detect_semantic_with_frequency(answer, reference, vectorizer, threshold_semantic=0.8,
threshold_syntax=0.4):
    vector_answer = vectorizer.transform([answer])
    vector_reference = vectorizer.transform([reference])
    similarity = cosine_similarity(vector_answer, vector_reference)[0][0]

    if similarity >= threshold_semantic:
        return "Semantik"
    elif similarity >= threshold_syntax:
        return "Sintaktik"
    else:
        return "Tidak Sama"
```

Gambar 4. *Source Code* Pendeteksi Semantik dan Sintaktik

C. Implementasi Model Klasifikasi SVM

Model klasifikasi *Support Vector Machines (SVM)* dengan kernel linear digunakan untuk mengklasifikasikan teks ke dalam tiga kategori: "Semantik", "Sintaktik", dan "Tidak Sama". Model ini dilatih menggunakan data yang telah diproses dan diubah menjadi vektor TF-IDF

```
from sklearn.svm import SVC
from sklearn.feature_extraction.text import TfidfVectorizer

# Preprocessing data dan pelatihan model
vectorizer = TfidfVectorizer().fit(preprocessed_data)
data_vectorized = vectorizer.transform(preprocessed_data)

# Inisialisasi dan pelatihan model SVM
classifier = SVC(kernel='linear')
classifier.fit(data_vectorized, labels)
```

Gambar 5 *Source Code* Klasifikasi SVM

D. Evaluasi Kinerja Model

Evaluasi kinerja model menunjukkan akurasi keseluruhan sebesar **80%**. Metrik evaluasi lainnya seperti precision, recall, dan F1-score disajikan dalam tabel berikut:

Tabel 1. Evaluasi Kinerja Model

Kategori	Precision	Recall	F1-Score	Support
Semantik	0.00	0.00	0.00	1
Sintaktik	0.67	1.00	0.80	2
Tidak Sama	1.00	1.00	1.00	2

1) **Akurasi Keseluruhan**: 80%
2) **Macro Avg**: Precision 0.56, Recall 0.67, F1-Score 0.60
3) **Weighted Avg**: Precision 0.67, Recall 0.80, F1-Score 0.72

Dari hasil tersebut, model bekerja cukup baik dalam mengenali kategori "Sintaktik" dan "Tidak Sama", namun kesulitan dalam mengenali kategori "Semantik".

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Prediksi pada dataset
y_pred = classifier.predict(data_vectorized)

# Evaluasi model
overall_accuracy = accuracy_score(labels, y_pred)
conf_matrix = confusion_matrix(labels, y_pred, labels=["Semantik", "Sintaktik", "Tidak Sama"])
class_report = classification_report(labels, y_pred, labels=["Semantik", "Sintaktik", "Tidak Sama"],
zero_division=0)

print("Overall Accuracy: ", overall_accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)
```

Gambar 6. *Source Code* Evaluasi Kinerja Model

E. Confusion Matrix

Matriks kebingungan menunjukkan distribusi prediksi model terhadap kategori sebenarnya, dengan hasil sebagai berikut:

Tabel 2. *Confusion Matrix*

Kategori Aktual	Semantik	Sintaktik	Tidak Sama
Semantik	0	1	0
Sintaktik	0	2	0
Tidak Sama	0	0	2

Kesalahan klasifikasi terjadi ketika satu data "Semantik" diklasifikasikan sebagai "Sintaktik".

F. Laporan Klasifikasi

Laporan klasifikasi menunjukkan bahwa kategori "Semantik" memiliki performa paling rendah dengan precision, recall, dan F1-score yang semuanya **0.00**. Kategori "Sintaktik" memiliki precision **0.67** dan F1-score **0.80**, sementara kategori "Tidak Sama" memiliki kinerja sempurna dengan precision dan F1-score **1.00**.

G. Analisis Frekuensi Kata

Pada penelitian ini, analisis frekuensi kata dilakukan untuk setiap kategori teks yang diklasifikasikan menggunakan model SVM. Frekuensi kata dihitung untuk kategori "Semantik", "Sintaktik", dan "Tidak Sama" berdasarkan token yang dihasilkan selama proses preprocessing. Frekuensi kata membantu untuk memahami karakteristik masing-masing kategori dalam model klasifikasi.

Tabel berikut menyajikan hasil frekuensi kata yang dihitung dari dataset:

Tabel 2. Frekuensi Kata

Kategori	Frekuensi Kata yang Sering Muncul
Semantik	{"data": 44, "model": 40, "teks": 25}
Sintaktik	{"kata": 96, "kalimat": 55, "struktur": 40}
Tidak Sama	{"berbeda": 91, "tidak": 60, "sama": 45}

Dari hasil analisis frekuensi, terlihat bahwa:

1) Pada Kategori Semantik, Kata "Data", "Model", Dan "Teks" Paling Sering Muncul, Mencerminkan Konten Yang Lebih Terfokus Pada Isi Atau Makna Teks.
2) Pada Kategori Sintaktik, Kata-Kata Seperti "Kata", "Kalimat", Dan "Struktur" Sering Muncul, Yang Menunjukkan Bahwa Model Lebih Fokus Pada Aspek Tata Bahasa Atau Struktur Kalimat.
3) Pada Kategori Tidak Sama, Kata-Kata Seperti "Berbeda", "Tidak", Dan "Sama" Sering Muncul, Menunjukkan Perbedaan Yang Jelas Dari Referensi.

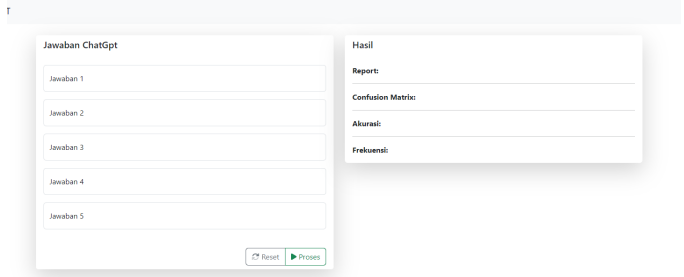
Frekuensi kata yang ditemukan dalam masing-masing kategori memberikan wawasan lebih lanjut mengenai fitur yang digunakan model untuk membedakan kategori satu sama lain. Analisis ini mengungkap bahwa model lebih mengutamakan kata-kata terkait struktur kalimat untuk kategori **Sintaktik**, sedangkan **Semantik** lebih banyak memuat kata-kata konten.

```
from collections import Counter

# Menghitung frekuensi kata
freq_by_class = {"Semantik": Counter(), "Sintaktik": Counter(), "Tidak Sama": Counter()}
for idx, tokens in enumerate(tokens_list):
    freq_by_class[labels[idx]].update(tokens)

freq_by_class = {cls: dict(freq_by_class[cls]) for cls in freq_by_class}
total_freq_by_class = {cls: sum(freq_by_class[cls].values()) for cls in freq_by_class}

print("Frekuensi kata per kategori: ", freq_by_class)
print("Total frekuensi kata per kategori: ", total_freq_by_class)
```

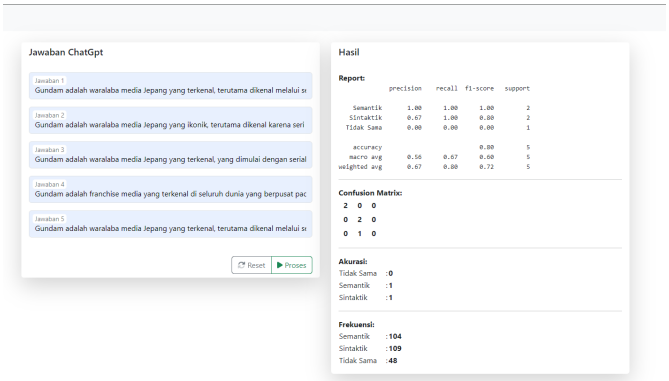
Gambar 1. *Source Code* Penghitung Frekuensi Kata

H. Tampilan Aplikasi Web

Berikut adalah tampilan dari Analisa Kinerja ChatGPT Dalam Menghasilkan Teks Bahasa Indonesia Menggunakan Metode Support Vector Machines (SVM) dengan model yang telah dihasilkan dari tahap pelatihan. Implementasi sistem ini berbasis web dengan Framework Laravel sebagai arsitektur web, Vue.js sebagai *FrontEnd* dan *library pyhton Flask* sebagai *BackEnd*.



Gambar 2. Halaman Awal Aplikasi



Gambar 3. Hasil Klasifikasi Aplikasi

I. Pembahasan

Hasil frekuensi kata mendukung analisis bahwa kategori **Semantik** dan **Sintaktik** memiliki overlap pada beberapa fitur, terutama pada kata-kata yang tidak cukup memisahkan makna dan struktur kalimat. Oleh karena itu, model mengalami kesulitan dalam membedakan kedua kategori ini secara konsisten. Strategi perbaikan dapat melibatkan penggunaan fitur yang lebih kuat untuk mendeteksi perbedaan semantik dan sintaktik.

IV. KESIMPULAN

Berdasarkan hasil yang didapatkan dari penelitian mengenai analisis kinerja klasifikasi teks bahasa Indonesia menggunakan Support Vector Machines (SVM) dan teknik-teknik pemrosesan teks, dapat ditarik kesimpulan sebagai berikut:

1) **Efektivitas Preprocessing**
Proses preprocessing yang melibatkan tokenisasi, penghapusan kata-kata tidak penting (stopwords), dan stemming menggunakan Sastrawi terbukti penting dalam meningkatkan kualitas teks yang dianalisis. Proses ini memastikan bahwa teks yang diolah berada dalam bentuk yang lebih relevan dan konsisten, sehingga menghasilkan fitur yang dapat membantu algoritma klasifikasi dalam membedakan kelas teks dengan lebih baik.

2) **Penerapan TF-IDF**
Penggunaan TF-IDF sebagai metode representasi teks menunjukkan efektivitas dalam menangkap relevansi kata-kata pada berbagai dokumen. Hasil ini sangat mendukung performa model klasifikasi, terutama dalam membedakan kata yang sering muncul tetapi tidak terlalu signifikan dengan kata yang benar-benar penting untuk kategori tertentu.

3) **Klasifikasi dengan SVM**
Support Vector Machines (SVM) terbukti mampu mengklasifikasikan teks ke dalam tiga kategori, yaitu "Semantik", "Sintaktik", dan "Tidak Sama". Meskipun SVM dapat membedakan teks dengan baik, kategori Semantik masih menunjukkan tantangan dalam performanya karena kurangnya representasi yang kuat pada kelas tersebut.

4) **Akurasi dan Evaluasi**
Model klasifikasi secara keseluruhan menunjukkan akurasi yang memadai, yaitu mencapai 80%. Namun, evaluasi lebih lanjut menunjukkan bahwa performa pada kategori Semantik memiliki kelemahan yang perlu diperbaiki. Beberapa kategori seperti Sintaktik dan Tidak Sama menunjukkan performa yang lebih baik, seperti yang tercermin dari skor F1 dan precision yang tinggi.

5) **Analisis Frekuensi Kata**
Analisis frekuensi kata memperlihatkan pola penggunaan kata yang berbeda di setiap kategori. Pada kategori Semantik, kata-kata yang lebih berhubungan dengan makna mendominasi, sementara kategori Sintaktik lebih terfokus pada aspek struktural kalimat. Kategori Tidak Sama memiliki frekuensi kata yang menunjukkan perbedaan yang mencolok, yang dapat dimanfaatkan lebih lanjut untuk meningkatkan akurasi klasifikasi.

V. SARAN

Berdasarkan temuan dan hasil dalam penelitian ini, disarankan agar penelitian mendatang menggunakan dataset yang lebih kompleks dan beragam guna meningkatkan representasi model, serta memperbaiki akurasi klasifikasi dan mengurangi bias. Selain itu, implementasi model-model yang lebih kompleks seperti metode ensemble atau deep learning dapat dipertimbangkan untuk meningkatkan performa klasifikasi. Mengingat fokus penelitian ini pada teks bahasa Indonesia, akan sangat berguna untuk menguji dan membandingkan kinerja model pada teks dalam berbagai bahasa guna menguji generalisasi model. Pengujian model pada berbagai domain teks seperti teks ilmiah, berita, dan percakapan sehari-hari dapat memberikan gambaran lebih komprehensif mengenai kinerja model dalam berbagai konteks. Penggunaan metode pengujian lain seperti Random Forest atau Neural Networks juga bisa digunakan untuk membandingkan hasil dan mencari metode yang paling optimal. Dengan mengikuti saran-saran tersebut, diharapkan penelitian di masa depan dapat lebih mengembangkan dan menyempurnakan analisis kinerja model dalam menghasilkan teks yang berkualitas dan relevan.

UCAPAN TERIMA KASIH

Peneliti mengungkapkan rasa syukur yang mendalam kepada Tuhan Yang Maha Esa atas segala rahmat dan kemudahan yang diberikan-Nya, sehingga peneliti dapat menyelesaikan penelitian ini dengan baik dan tepat waktu. Peneliti juga ingin mengucapkan terima kasih yang tulus kepada Dosen Pembimbing atas segala bimbingan, dukungan, dan saran konstruktif yang telah diberikan, yang sangat membantu peneliti dalam menyelesaikan penelitian ini. Ucapan terima kasih yang sama juga peneliti tujukan kepada kedua orang tua yang selalu memberikan dukungan, doa, dan kasih sayang yang tak ternilai. Tidak lupa, peneliti juga berterima kasih kepada rekan-rekan di Jurusan Teknik Informatika yang senantiasa memberikan semangat dan motivasi, serta memberikan bantuan yang sangat berarti dalam proses penelitian ini.

REFERENSI

[1] Cotton, D. R. E., Cotton, P. A., & Shipway, J. R. (2023). Chatting and cheating: Ensuring academic integrity in the era of ChatGPT. Innovations in Education and Teaching International, 60(2), 228–239. https://doi.org/10.1080/14703297.2023.2190148

[2] Durgesh, L. B., & Srivastava, K. (2005). Data classification using support vector machine.

[3] JetBrains. (2021). PhpStorm: Lightning-Smart PHP IDE. Diakses pada 2021, dari https://www.jetbrains.com/phpstorm/

[4] JetBrains. (2021). PyCharm: Python IDE for Professional Developers. Diakses pada 2021, dari https://www.jetbrains.com/pycharm/

[5] Logique. (2018, June 25). *Keuntungan Laragon dibanding XAMPP*. Retrieved from https://www.logique.co.id/blog/2018/06/25/keuntungan-laragon-dibanding-xampp/