

ADVANCED NLP

IWAN MUTTAQIN

231012050010



Ebook ini dibuat untuk penyelesaian Tugas Mingguan
Advance NLP pada Semester 3 Pascasarjana Teknik
Informatika Universitas Pamulang | 2025

Dosen Pengampu :

DR. IR AGUNG BUDI SUSANTO MM

Dr. SAJARWO ANGGAI S.ST., M.T



Buku ini menyajikan panduan komprehensif untuk menguasai Natural Language Processing (NLP) tingkat lanjut, mulai dari konsep dasar hingga implementasi di dunia nyata. Dengan pendekatan terstruktur, pembaca akan diajak menjelajahi seluruh tahapan pipeline NLP—dimulai dari pengenalan definisi, sejarah, dan tantangan NLP, hingga teknik modern berbasis deep learning dan transformers.

Target Pembaca:

- ✓ *Pemula yang ingin memahami NLP dari dasar.*
 - ✓ *Practitioner (insinyur ML, data scientist) yang membutuhkan panduan implementasi.*
 - ✓ *Akademisi/peneliti yang tertarik eksplorasi tantangan terkini di NLP.*

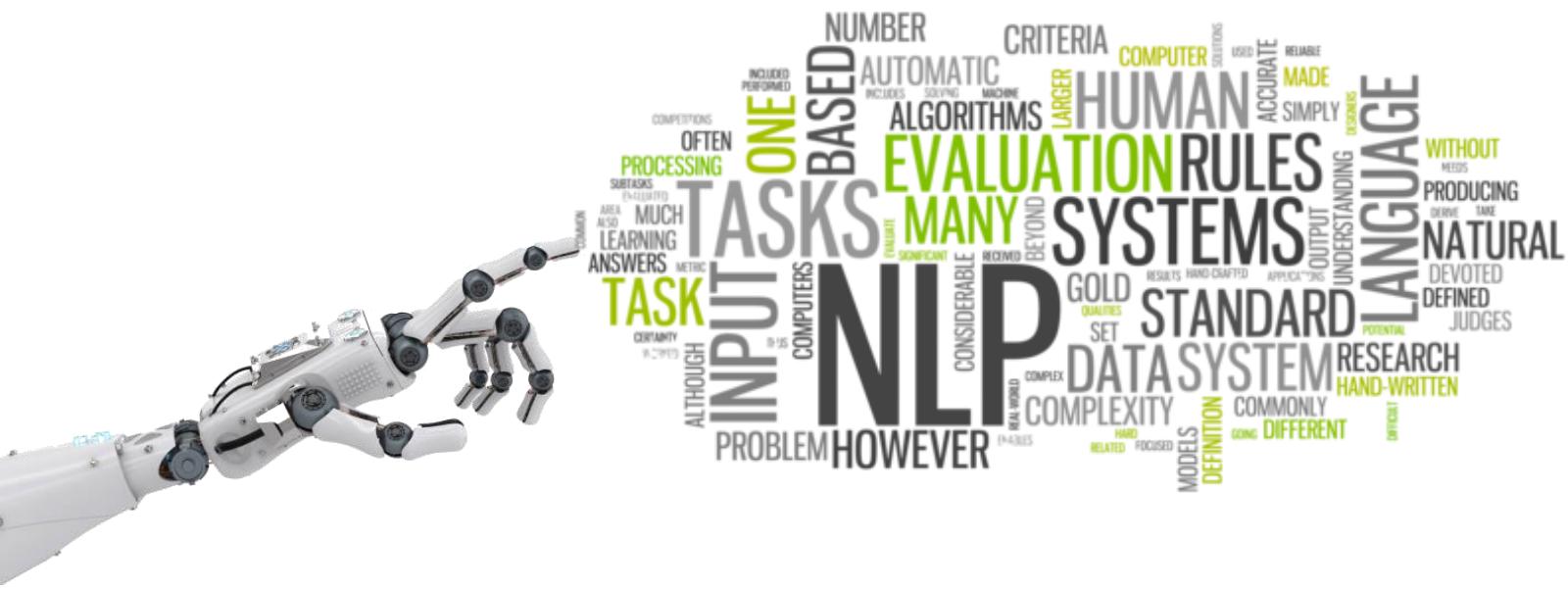
Nilai Tambah:

- ✓ Kombinasi teori mendalam dan panduan praktis dengan kode Python.
 - ✓ Pembahasan tantangan NLP spesifik bahasa Indonesia.
 - ✓ Proyek akhir untuk mengonsolidasi pembelajaran (end-to-end NLP system).

Buku ini dirancang sebagai referensi lengkap untuk membangun solusi NLP yang robust, sekaligus memahami state-of-the-art teknologi di balik sistem seperti ChatGPT dan Google Translate.

Download Full Source Code [disini](#):

<https://github.com/iwancilibur/BELAJAR-NLP>



DAFTAR ISI

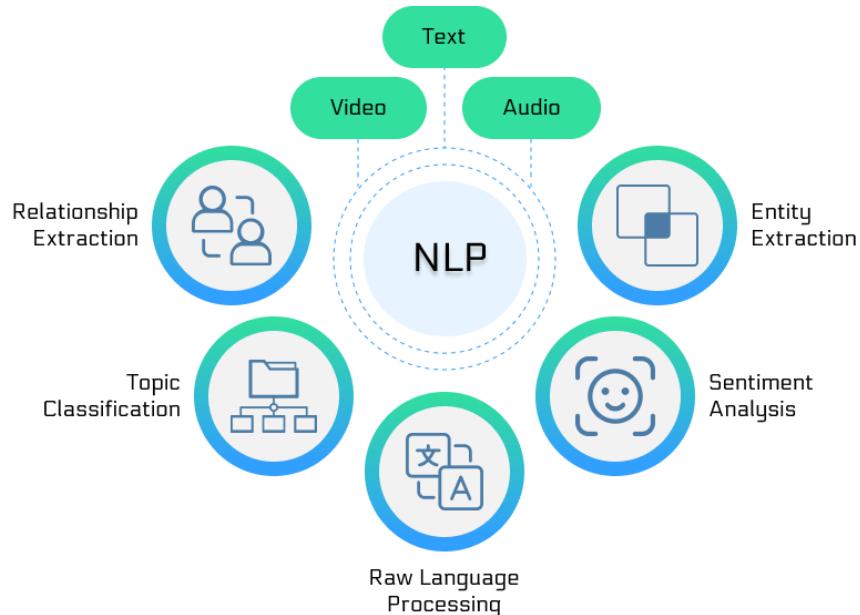
DAFTAR ISI.....	3
PERTEMUAN I INTRODUCTION NLP	4
1.1. Apa itu NLP ?	4
1.2. Referensi Jurnal Terkait NLP dan Link yang dapat diakses.....	6
1.3. Perusahaan / Instansi Terkait yang Menggunakan NLP	7
PERTEMUAN II PYTHON LIBRARY FOR NLP	9
2.1. Python <i>Libraries</i> For NLP	9
2.2. Jurnal yang Menggunakan Bahasa Python Sebagai <i>Tools</i> NLP	12
2.3. Perusahaan / Instansi terkait yang menggunakan Python dan NLP.....	13
PERTEMUAN III TEXT PREPROCESSING.....	15
3.1. Apa itu Text Preprocessing.....	15
3.2. Jurnal Terkait Peggunaan Text Preprocessing pada NLP.....	16
3.3. Program Codelabs Penggunaan Text Preprocessing pada NLP	16
PERTEMUAN IV FEATURE EXTRACTION.....	18
4.1. Apa itu <i>Feature Extraction</i>	18
4.2. Jurnal Terkait Penggunaan <i>Feature Extraction</i> pada NLP.....	19
4.3. Program Codelabs Penggunaan <i>Feature Extraction</i> pada NLP	20
PERTEMUAN V INFORMATION RETRIEVAL	21
5.1. Apa itu <i>Infromation Retrieval</i>	21
5.2. Jurnal Terkait Penggunaan <i>Infromation Retrieval</i> pada NLP	21
5.3. Program Codelabs Penggunaan <i>Infromation Retrieval</i> pada <i>NLP</i> Sederhana.....	22
5.4. Program Codelabs Penggunaan <i>Inverted Index</i> untuk menampung corpus!	24
PERTEMUAN VI TOPIK MODEL MENGGUNAKAN LDA	28
6.1. Apa itu Topik Model Menggunakan LDA	28
6.2. Jurnal Terkait Penggunaan Topic Model (LDA dan Turunannya)	28
6.3. Program Codelabs Model LDA (Datasheet Bahasa Indonesia dan Inggris)	29
PERTEMUAN VII TOPIK MODEL MENGGUNAKAN BERT.....	33
7.1. Apa itu Topik Model Menggunakan BERT	33
7.2. Jurnal Terkait Penggunaan Topic Model BERT	33
7.3. Program Codelabs Menampilkan Hasil Model BERTopic	34
DAFTAR PUSTAKA	38

PERTEMUAN I

INTRODUCTION NLP

1.1. Apa itu NLP ?

Natural Language Processing (NLP) adalah cabang ilmu *Artificial Intelligence* (AI) dan *linguistik komputasional* yang fokus pada interaksi antara komputer dan bahasa manusia. Tujuannya adalah memungkinkan mesin memahami, memproses, dan menghasilkan bahasa alami seperti manusia.



Sumber : <https://amazinum.com/insights/what-is-nlp-and-how-it-is-implemented-in-our-lives>

A. Contoh Aplikasi NLP:

- *Chatbots* (contoh: ChatGPT, Google Assistant).
- *Machine Translation* (contoh: Google Translate).
- *Sentiment Analysis* (analisis opini di media sosial).
- *Text Summarization* (ringkasan dokumen otomatis).

B. Cara Kerja NLP:

Pemrosesan bahasa alami (NLP) menggabungkan model linguistik komputasional, *machine learning*, dan *deep learning* untuk memproses bahasa manusia.

1) Linguistik komputasional

Linguistik komputasional adalah ilmu memahami dan membangun model bahasa manusia dengan alat komputer dan perangkat lunak. Para peneliti menggunakan metode linguistik komputasional, seperti analisis sintaksis dan semantik, untuk

menciptakan kerangka kerja yang membantu mesin memahami bahasa manusia yang digunakan dalam percakapan. Alat seperti penerjemah bahasa, *synthesizer teks ke ucapan*, dan perangkat lunak pengenalan ucapan didasarkan pada linguistik komputasional.

2) *Machine learning*

Machine learning adalah teknologi yang melatih komputer dengan data sampel untuk meningkatkan efisiensinya. Bahasa manusia memiliki sejumlah fitur seperti sarkasme, metafora, variasi dalam struktur kalimat, serta tata bahasa dan pengecualian penggunaan yang memerlukan waktu bertahun-tahun untuk dipelajari oleh manusia. *Programmer* menggunakan metode *machine learning* untuk mengajari aplikasi NLP mengenali dan memahami fitur-fitur ini secara akurat sejak awal.

3) Deep learning

Deep learning adalah sebuah bidang *machine learning* spesifik yang mengajari komputer untuk belajar dan berpikir seperti manusia. *Deep learning* melibatkan jaringan neural yang terdiri dari simpul pemrosesan data yang terstruktur untuk menyerupai operasi otak manusia. Dengan *deep learning*, komputer mengenali, menglasifikasikan, dan menghubungkan pola kompleks dalam data input.

C. Langkah-langkah implementasi NLP

Biasanya, implementasi NLP dimulai dengan mengumpulkan dan menyiapkan data teks atau ucapan yang tidak terstruktur dari banyak sumber seperti gudang data *cloud*, survei, *email*, atau aplikasi proses bisnis internal.

1) *Prapemrosesan*

Perangkat lunak NLP menggunakan teknik prapemrosesan seperti tokenisasi, *stemming*, lemmatisasi, dan penghapusan kata henti guna menyiapkan data untuk berbagai aplikasi.

Berikut deskripsi teknik-teknik ini:

- Tokenisasi memecah sebuah kalimat menjadi unit kata atau frasa individual.
- *Stemming* dan lemmatisasi menyederhanakan kata ke dalam bentuk akarnya. Misalnya, proses ini mengubah “*starting*” menjadi “*start*”.

- Penghapusan kata henti memastikan bahwa kata yang tidak menambahkan makna signifikan ke sebuah kalimat, seperti “for” dan “with”, dihapus.

2) *Pelatihan*

Para peneliti menggunakan data yang diproses sebelumnya dan *machine learning* untuk melatih model NLP guna menjalankan aplikasi spesifik berdasarkan informasi tekstual yang disediakan. Pelatihan algoritma NLP memerlukan pemberian sampel data besar pada perangkat lunak untuk meningkatkan akurasi algoritma.

3) *Deployment dan inferensi*

Ahli *machine learning* kemudian melakukan *deployment* model atau mengintegrasikan model tersebut ke dalam lingkungan produksi yang sudah ada. Model NLP menerima input dan memprediksi *output* untuk kasus penggunaan spesifik yang didesain untuk model tersebut. Anda dapat menjalankan aplikasi NLP di data langsung dan mendapatkan *output* yang diperlukan.

1.2. Referensi Jurnal Terkait NLP dan Link yang dapat diakses

No	Nama Penulis & Tahun	Judul	Model	Hasil Penelitian
1	Vaswani et al. (2017)	<u>Attention Is All You Need</u>	Transformer	Memperkenalkan arsitektur Transformer yang menjadi dasar model modern seperti GPT dan BERT.
2	Devlin et al. (2019)	<u>BERT: Pre-training of Deep Bidirectional Transformers</u>	BERT	Mencapai SOTA pada berbagai tugas NLP dengan pre-training bidirectional.
3	Radford et al. (2018)	<u>Improving Language Understanding by Generative Pre-Training</u>	GPT-1	Memperkenalkan pendekatan generative pre-training untuk NLP.
4	Brown et al. (2020)	<u>Language Models are Few-Shot Learners</u>	GPT-3	Menunjukkan kemampuan generalisasi model bahasa skala besar dengan sedikit contoh.

5	Mikolov et al. (2013)	<u>Efficient Estimation of Word Representations in Vector Space</u>	Word2Vec	Mengusulkan embedding kata yang efisien untuk representasi semantik.
6	Pennington et al. (2014)	<u>GloVe: Global Vectors for Word Representation</u>	GloVe	Menghasilkan vektor kata berbasis matriks co-occurrence global.
7	Hochreiter & Schmidhuber (1997)	<u>Long Short-Term Memory</u>	LSTM	Arsitektur RNN yang efektif menangani ketergantungan jarak jauh.
8	Conneau et al. (2017)	<u>Supervised Learning of Universal Sentence Representations</u>	InferSent	Membuat embedding kalimat universal dengan supervised learning.
9	Joulin et al. (2017)	<u>Bag of Tricks for Efficient Text Classification</u>	FastText	Klasifikasi teks cepat dengan n-gram karakter dan hierarki softmax.
10	Liu et al. (2019)	<u>RoBERTa: A Robustly Optimized BERT Pretraining Approach</u>	RoBERTa	Optimasi BERT dengan pelatihan lebih lama dan data lebih besar.

1.3. Perusahaan / Instansi Terkait yang Menggunakan NLP

- 1) **Chatbots dan Virtual Assistants** seperti ChatGPT, Alexa, Google Assistant, dan Bot Tokopedia telah banyak diimplementasikan dalam sektor e-commerce, layanan publik, dan perbankan. Teknologi ini dimanfaatkan untuk mengotomatiskan layanan pelanggan dan menyediakan sistem tanya jawab yang cepat, meningkatkan efisiensi interaksi antara pengguna dan sistem.
- 2) **Machine Translation** yang diwakili oleh layanan seperti Google Translate, DeepL, dan Microsoft Translator berperan penting dalam sektor pendidikan, pemerintahan, dan media. Teknologi ini memfasilitasi komunikasi lintas bahasa, memungkinkan pertukaran informasi yang lebih inklusif dan global.
- 3) **Speech-to-Text dan Text-to-Speech**, dengan contoh seperti Google Speech API dan IBM Watson TTS, telah digunakan dalam industri otomotif, call center, dan sektor kesehatan. Teknologi ini berguna untuk transkripsi otomatis dan menyediakan antarmuka suara sebagai asisten yang responsif dan mudah diakses.
- 4) **Sentiment Analysis** atau analisis sentimen diterapkan pada ulasan pelanggan seperti di Amazon, Twitter, dan media sosial lainnya. Digunakan dalam sektor e-commerce,

pemasaran, dan politik, teknologi ini membantu dalam mengukur opini publik atau persepsi pelanggan terhadap produk atau isu tertentu.

- 5) **Text Classification** digunakan dalam berbagai aplikasi seperti spam filter Gmail dan moderasi konten di Facebook. Sektor yang memanfaatkannya mencakup keamanan digital, media sosial, dan layanan email, dengan manfaat utama berupa deteksi spam, identifikasi konten ofensif, serta pengelompokan dokumen relevan secara otomatis.
- 6) **Information Extraction**, terutama Named Entity Recognition (NER) dan Relation Extraction (RE), digunakan dalam analisis kontrak hukum dan berita otomatis. Sektor hukum, keuangan, dan riset mengandalkan teknologi ini untuk mengekstraksi entitas penting dan informasi spesifik dari dokumen teks.
- 7) **Document Summarization** atau peringkasan dokumen, seperti yang diterapkan oleh Bloomberg News Summarizer dan dalam ringkasan dokumen hukum, telah menjadi solusi di sektor jurnalistik, hukum, dan penelitian. Teknologi ini mempercepat pemrosesan informasi dari dokumen panjang secara efisien.
- 8) **Question Answering Systems**, contohnya Bing Copilot dan chatbot instansi pemerintah, mendukung sektor pemerintahan, edukasi, dan layanan publik. Sistem ini memungkinkan penyajian jawaban cepat dan akurat dari dokumen atau basis data untuk menjawab pertanyaan pengguna.
- 9) **OCR (Optical Character Recognition) + NLP** telah diterapkan dalam pengolahan e-KTP dan dokumen pajak. Dalam sektor pemerintahan, keuangan, dan kesehatan, teknologi ini bermanfaat untuk mengekstraksi data dari dokumen fisik ke bentuk digital secara otomatis.
- 10) **Legal & Compliance NLP**, termasuk untuk keperluan Anti Money Laundering dan analisis kontrak otomatis, sangat berguna di sektor keuangan, hukum, dan pemerintahan. Manfaat utamanya adalah untuk memastikan kepatuhan hukum secara otomatis serta mendeteksi potensi risiko hukum dari dokumen atau transaksi.

PERTEMUAN II

PYTHON LIBRARY FOR NLP

2.1. Python Libraries For NLP

Berikut adalah library Python populer yang digunakan dalam NLP:

1) NLTK (Natural Language Toolkit)

Fungsi: Tokenisasi, stemming, POS tagging.

Contoh:

```
1. import nltk  
2. nltk.download('punkt')  
3. tokens = nltk.word_tokenize("Hello, NLP!")
```

Tokenisasi: Memecah teks menjadi kata, kalimat, atau simbol (token).

```
1. from nltk.tokenize import word_tokenize  
2. tokens = word_tokenize("Hello, world!") # Output: ['Hello', ',', 'world', '!']  
3.
```

Stemming & Lemmatization: Mengurangi kata ke bentuk dasar.

```
1. from nltk.stem import PorterStemmer  
2. stemmer = PorterStemmer()  
3. stemmer.stem("running") # Output: 'run'
```

POS Tagging: Menandai kelas kata (kata kerja, kata benda, dll.).

```
1. nltk.pos_tag(["NLP", "is", "awesome"]) # Output: [('NLP', 'NNP'), ('is', 'VBZ'), ('awesome', 'JJ')]
```

Stopword Removal: Menghapus kata umum yang tidak informatif

(contoh: "the", "and").

Chunking/NER: Mendeteksi entitas bernama (nama orang, lokasi).

Kelebihan:

- Cocok untuk pemula dan riset akademik.
- Mendukung banyak bahasa.

Kekurangan:

- Lambat untuk data besar.

2) spaCy

Fungsi: Pemrosesan teks cepat dengan dukungan deep learning.

Contoh:

```
1. import spacy
2. nlp = spacy.load("en_core_web_sm")
3. doc = nlp("NLP is amazing.")
```

Pipeline NLP Terintegrasi: Tokenisasi, POS tagging, NER, parsing dependensi dalam satu eksekusi cepat.

```
1. import spacy
2. nlp = spacy.load("en_core_web_sm")
3. doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
4. for ent in doc.ents:
5.     print(ent.text, ent.label_) # Output: Apple ORG, U.K. GPE, $1 billion MONEY
```

Word Vectors: Embedding kata berbasis konteks (tersedia dalam model md atau lg).

Custom Pipeline: Membuat model NER atau text classifier sendiri.

Kelebihan:

- Cepat dan optimal untuk produksi.
- Dukungan deep learning (integrasi dengan PyTorch/TensorFlow).

Kekurangan:

- Model bahasa besar membutuhkan banyak memori.

3) Transformers (Hugging Face)

Fungsi: Implementasi model BERT, GPT, dll.

Contoh:

```
1. from transformers import pipeline
2. classifier = pipeline("sentiment-analysis")
```

Pre-trained Models: Menggunakan model SOTA seperti BERT, GPT, T5.

```
1. from transformers import pipeline
2. classifier = pipeline("sentiment-analysis")
3. classifier("I love NLP!") # Output: [{'label': 'POSITIVE', 'score': 0.99}]
```

Fine-tuning: Melatih ulang model untuk tugas spesifik (contoh: klasifikasi teks).

Tokenizers Cepat: Mendukung tokenisasi subword (misal: WordPiece untuk BERT).

Kelebihan:

- Akses ke ribuan model pra-pelatihan.
- Kompatibel dengan PyTorch dan TensorFlow.

Kekurangan:

- Membutuhkan GPU untuk pelatihan skala besar.

4) Gensim

Fungsi: Topic modeling (LDA, Word2Vec).

Topic Modeling: Implementasi LDA, LSI untuk ekstraksi topik.

```
1. from gensim.models import LdaModel  
2. lda = LdaModel(corpus=corpus, num_topics=5) # 5 topik
```

Word Embeddings: Word2Vec, FastText, Doc2Vec.

```
1. from gensim.models import Word2Vec  
2. model = Word2Vec(sentences, vector_size=100, window=5)
```

Similarity Search: Mencari dokumen/kata mirip menggunakan cosine similarity.

Kelebihan:

- Efisien untuk dokumen besar.
- Fokus pada unsupervised learning.

Kekurangan:

- Tidak mendukung tugas NLP kompleks seperti NER.

5) extBlob

Fungsi: Analisis sentimen sederhana.

Sentiment Analysis: Analisis sentimen sederhana (polaritas dan subjektivitas).

```
1. from textblob import TextBlob  
2. blob = TextBlob("NLP is amazing.")  
3. print(blob.sentiment) # Output: Sentiment(polarity=0.8, subjectivity=0.75)
```

Terjemahan & Spell Check:

```
1. blob.translate(to="es") # Terjemahkan ke Spanyol  
2. blob.correct() # Koreksi ejaan
```

Kelebihan:

- Sintaks mudah untuk pemula.
- Dibangun di atas NLTK dan Pattern.

Kekurangan:

- Terbatas untuk tugas kompleks.

Perbandingan Singkat

Library	Keunggulan	Kekurangan	Use Case
NLTK	Lengkap untuk riset	Lambat	Edukasi, Eksperimen
spaCy	Cepat, produksi-ready	Model bahasa besar	Industri, Aplikasi Real-time
Transformers	Model SOTA, fleksibel	Butuh sumber daya tinggi	Fine-tuning BERT/GPT
Gensim	Efisien untuk dokumen besar	Tidak untuk NLP kompleks	Topic Modeling, Embedding
TextBlob	Mudah digunakan	Fitur terbatas	Analisis Sentimen Sederhana

Tips Pemilihan Library:

- Gunakan **spaCy** untuk pipeline NLP end-to-end.
- Pilih **Transformers** jika butuh model deep learning.
- **NLTK** cocok untuk pembelajaran konsep dasar.

2.2.Jurnal yang Menggunakan Bahasa Python Sebagai *Tools* NLP

No	Nama Penulis & Tahun	Judul	Model	Hasil Penelitian
1	Loper & Bird (2002)	NLTK: The Natural Language Toolkit	NLTK	Library Python untuk pendidikan dan penelitian NLP.
2	Honnibal & Montani (2017)	spaCy: Industrial-Strength NLP	spaCy	Library efisien untuk pemrosesan teks produksi.
3	Wolf et al. (2020)	Transformers: State-of-the-Art NLP	Transformers	Integrasi model SOTA seperti BERT dan GPT.
4	Řehůřek & Sojka (2010)	Gensim: Topic Modelling for Humans	LDA, Word2Vec	Implementasi mudah untuk topic modeling.
5	Pedregosa et al. (2011)	Scikit-learn: Machine Learning in Python	TF-IDF, SVM	Library serbaguna untuk ML termasuk NLP.
6	Bird et al. (2009)	Natural Language Processing with Python	NLTK	Panduan komprehensif NLP menggunakan Python.
7	Qiu et al. (2020)	Pre-trained Models for NLP	BERT, GPT	Survey model pre-trained berbasis Python.

8	Akbik et al. (2018)	<u>Contextual String Embeddings</u>	Flair	Embedding kontekstual untuk NER dan klasifikasi.
9	Zhang et al. (2019)	<u>Peeling the Onion: Hierarchical Layerwise NLP</u>	PyTorch	Analisis lapisan dalam model NLP.
10	Paszke et al. (2019)	<u>PyTorch: An Imperative Style for NLP</u>	PyTorch	Framework deep learning fleksibel untuk NLP.

2.3. Perusahaan / Instansi terkait yang menggunakan Python dan NLP

- 1) **Text Preprocessing Tools** seperti NLTK, spaCy, dan TextBlob digunakan secara luas di sektor pendidikan, riset, dan industri. Teknologi ini berfungsi untuk membersihkan dan menyiapkan data teks sebelum dilakukan analisis lebih lanjut, seperti menghapus tanda baca, stemming, tokenisasi, dan normalisasi kata, yang menjadi tahap awal penting dalam pipeline Natural Language Processing (NLP).
- 2) **Sentiment Analysis Tools** seperti TextBlob, VADER, dan pysentimiento banyak dimanfaatkan dalam sektor e-commerce, media sosial, dan layanan publik. Dengan teknologi ini, sistem dapat menilai opini atau sentimen pengguna terhadap produk, layanan, atau isu tertentu, sehingga membantu pengambilan keputusan berbasis data emosional atau persepsi publik.
- 3) **Named Entity Recognition (NER)** dengan pustaka seperti spaCy dan HuggingFace Transformers digunakan di bidang hukum, kesehatan, dan media digital. NER memungkinkan sistem untuk mendeteksi entitas penting dalam teks, seperti nama orang, lokasi, organisasi, dan tanggal, yang berguna dalam ekstraksi informasi dan pelabelan otomatis.
- 4) **Text Classification** dengan dukungan dari pustaka seperti Scikit-learn, FastText, dan Keras diterapkan dalam sektor pemasaran, email filtering, dan moderasi konten. Teknologi ini mengklasifikasikan teks secara otomatis ke dalam kategori tertentu, misalnya untuk mendeteksi spam, menentukan sentimen, atau mengidentifikasi topik dari suatu dokumen.
- 5) **Topic Modeling** menggunakan Gensim (dengan pendekatan LDA) dan BERTopic sering digunakan dalam penerbitan ilmiah dan riset pasar. Teknologi ini membantu mengelompokkan kumpulan teks ke dalam topik-topik utama, sehingga memudahkan analisis tema atau tren dalam dokumen yang besar.

- 6) **Machine Translation** dengan pustaka seperti OpenNMT, Fairseq, dan MarianMT mendukung sektor pendidikan, diplomasi, dan teknologi global. NLP dalam bentuk penerjemahan mesin ini memungkinkan konversi teks antar bahasa secara akurat dan cepat, sehingga memperlancar komunikasi lintas budaya dan negara.
- 7) **Chatbot Development** dengan framework seperti Rasa, ChatterBot, dan DeepPavlov telah diadopsi oleh sektor e-commerce, perbankan, dan layanan publik. Teknologi ini digunakan untuk membangun sistem chatbot cerdas yang dapat menjawab pertanyaan pengguna secara otomatis dan memberikan pelayanan pelanggan 24/7.
- 8) **Text Summarization** dengan model seperti BART, T5 (Transformers), dan pustaka Sumy, memberikan solusi di bidang media, hukum, dan edukasi. Teknologi ini memungkinkan sistem untuk merangkum dokumen atau artikel panjang menjadi ringkasan yang padat dan informatif, menghemat waktu pembaca.
- 9) **Question Answering Systems** yang memanfaatkan Haystack dan model BERT QA dari Transformers digunakan dalam pemerintahan dan sistem pencarian internal. Teknologi ini memungkinkan pencarian jawaban spesifik dari dokumen panjang atau basis data, sangat bermanfaat dalam otomatisasi layanan informasi.
- 10) **Legal & Compliance NLP**, seperti sistem untuk Anti Money Laundering dan analisis kontrak otomatis, berperan penting di sektor keuangan, hukum, dan pemerintahan. Teknologi ini membantu dalam mendeteksi risiko hukum, memastikan kepatuhan secara otomatis, serta mempermudah audit dan tinjauan kontrak secara efisien.

PERTEMUAN III

TEXT PREPROCESSING

3.1. Apa itu Text Preprocessing

Text preprocessing adalah serangkaian teknik untuk **membersihkan dan mengubah teks mentah** menjadi format yang siap diproses oleh model NLP. Tujuannya adalah:

- Menghilangkan noise (tanda baca, karakter khusus).
- Menstandarisasi format teks.
- Mengurangi kompleksitas tanpa kehilangan makna penting.

Tahapan Text Preprocessing

Berikut tahapan umum beserta contoh implementasinya:

1. Case Folding

Mengubah teks menjadi lowercase.

```
1. text = "NLP is AWESOME!"  
2. text.lower() # Output: "nlp is awesome!"
```

2. Tokenisasi

Memecah teks menjadi kata/token.

```
1. from nltk.tokenize import word_tokenize  
2. tokens = word_tokenize("I love NLP!") # Output: ['I', 'love', 'NLP', '!']
```

3. Filtering (Stopword Removal)

Menghapus kata umum yang tidak informatif.

```
1. from nltk.corpus import stopwords  
2. stop_words = set(stopwords.words('english'))  
3. filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
```

4. Stemming/Lemmatization

Mengurangi kata ke bentuk dasar.

```
1. from nltk.stem import PorterStemmer  
2. stemmer = PorterStemmer()  
3. stemmer.stem("running") # Output: "run"
```

5. Normalisasi

Mengganti singkatan/typo (contoh: "brb" → "be right back").

6. Menghapus Tanda Baca dan Angka

```
1. import re  
2. text = "Harga: Rp 10.000,-"  
3. re.sub(r'[^\w\s]', '', text) # Output: "Harga Rp "
```

3.2. Jurnal Terkait Peggunaan Text Preprocessing pada NLP

No	Nama Penulis & Tahun	Judul	Model	Hasil Penelitian
1	Loper & Bird (2002)	NLTK: Tokenization and Cleaning	Tokenisasi, Stopword Removal	Dasar-dasar preprocessing untuk riset NLP.
2	Jurafsky & Martin (2020)	Speech and Language Processing	Lemmatization, Normalisasi	Panduan komprehensif teknik preprocessing.
3	Bird et al. (2009)	Natural Language Processing with Python	Stemming, POS Tagging	Implementasi praktis dengan NLTK.
4	Aggarwal & Zhai (2012)	A Survey of Text Normalization	Normalisasi Teks	Analisis metode normalisasi untuk NLP.
5	Manning et al. (2008)	Introduction to Information Retrieval	Tokenisasi, Case Folding	Dampak preprocessing pada IR.
6	Mikolov et al. (2013)	Word2Vec: Preprocessing Matters	Filtering Karakter Khusus	Pengaruh preprocessing pada embedding.
7	Zhang et al. (2015)	Character-level CNN for Text	Tanpa Tokenisasi	Preprocessing minimal untuk CNN.
8	Joulin et al. (2016)	Bag of Tricks for Efficient Text Classification	N-gram, Hashing	Teknik preprocessing untuk klasifikasi cepat.
9	Howard & Ruder (2018)	ULMFiT: Universal Language Model Fine-tuning	Subword Tokenization	Preprocessing untuk transfer learning.
10	Devlin et al. (2019)	BERT: Preprocessing for Deep Learning	WordPiece Tokenization	Standar preprocessing untuk model Transformer.

3.3. Program Codelabs Penggunaan Text Preprocessing pada NLP

```

1. import re
2. import nltk
3. from nltk.tokenize import word_tokenize
4. from collections import Counter
5.
6. # Download semua resource yang diperlukan
7. nltk.download('punkt')
8. nltk.download('punkt_tab')
9.
10. # Stopword bahasa Indonesia (sederhana)
11. stopwords_id = {

```

```

12.     'yang', 'dan', 'di', 'ke', 'dari', 'ini', 'itu', 'atau',
13.     'tidak', 'dengan', 'untuk', 'pada', 'adalah', 'juga'
14. }
15.
16. def preprocess_text_id(text):
17.     # Case folding
18.     text = text.lower()
19.     # Hapus tanda baca/angka
20.     text = re.sub(r'[^a-zA-Z\s]', '', text)
21.     # Tokenisasi
22.     tokens = word_tokenize(text, language='english') # Gunakan tokenizer Inggris
23.     # Stopword removal
24.     tokens = [word for word in tokens if word not in stopwords_id]
25.
26.     # Hitung statistik
27.     stats = {
28.         'jumlah_kata': len(tokens),
29.         'kata_unik': len(set(tokens)),
30.         'kata_terbanyak': Counter(tokens).most_common(3),
31.         'daftar_kata': list(set(tokens))
32.     }
33.     return " ".join(tokens), stats
34.
35. # Contoh teks berita
36. berita = """
37. Pemerintah hari ini mengumumkan bantuan sosial sebesar Rp 10 triliun untuk UMKM.
38. Bantuan ini diberikan kepada pelaku usaha yang terdampak krisis ekonomi.
39. Menteri Keuangan mengatakan bantuan akan cair mulai bulan depan.
40. """
41.
42. # Proses teks
43. cleaned_text, hasil = preprocess_text_id(berita)
44.
45. # Hasil
46. print("== TEKS BERITA ASLI ==")
47. print(berita)
48.
49. print("\n== HASIL PREPROCESSING ==")
50. print(cleaned_text)
51.
52. print("\n== ANALISIS TEKS ==")
53. print(f"Jumlah kata total: {hasil['jumlah_kata']} ")
54. print(f"Jumlah kata unik: {hasil['kata_unik']} ")
55. print(f"3 kata terbanyak: {hasil['kata_terbanyak']} ")
56. print("Daftar kata unik: ", ", ".join(hasil['daftar_kata']))
```

Berikut Hasil yang akan ditampilkan:

```

== TEKS BERITA ASLI ==

Pemerintah hari ini mengumumkan bantuan sosial sebesar Rp 10 triliun untuk UMKM.
Bantuan ini diberikan kepada pelaku usaha yang terdampak krisis ekonomi.
Menteri Keuangan mengatakan bantuan akan cair mulai bulan depan.

== HASIL PREPROCESSING ==
pemerintah hari mengumumkan bantuan sosial sebesar rp triliun umkm bantuan diberikan kepada pelaku usaha terdampak
krisis ekonomi menteri keuangan mengatakan bantuan akan cair mulai bulan depan

== ANALISIS TEKS ==
Jumlah kata total: 26
Jumlah kata unik: 24
3 kata terbanyak: [('bantuan', 3), ('pemerintah', 1), ('hari', 1)]
Daftar kata unik: pelaku, bulan, menteri, kepada, triliun, mengatakan, ekonomi, mengumumkan, mulai, hari, keuangan,
akan, terdampak, bantuan, rp, sosial, umkm, cair, diberikan, pemerintah, sebesar, depan, krisis, usaha
```

PERTEMUAN IV

FEATURE EXTRACTION

4.1.Apa itu *Feature Extraction*

Feature Extraction adalah proses mengubah teks mentah menjadi representasi numerik yang bisa diproses oleh model machine learning. Tujuannya adalah menangkap karakteristik penting dari teks sambil mengurangi dimensi data.

Teknik Feature Extraction untuk NLP

Berikut teknik utama beserta implementasi Python:

1) *Bag-of-Words (BoW)*

Merepresentasikan teks sebagai vektor frekuensi kata.

```
1. from sklearn.feature_extraction.text import CountVectorizer
2. corpus = ["Saya suka NLP", "NLP itu menarik"]
3. vectorizer = CountVectorizer()
4. X = vectorizer.fit_transform(corpus)
5. print(vectorizer.get_feature_names_out()) # Output: ['itu', 'menarik', 'nlp',
'suka', 'saya']
```

2) *TF-IDF (Term Frequency-Inverse Document Frequency)*

Memberi bobot kata berdasarkan kepentingannya di dokumen vs korpus.

```
1. from sklearn.feature_extraction.text import TfidfVectorizer
2. tfidf = TfidfVectorizer()
3. X = tfidf.fit_transform(corpus)
```

3) *Word Embeddings*

Representasi kata dalam vektor padat (contoh: Word2Vec, GloVe).

```
1. from gensim.models import Word2Vec
2. sentences = [["saya", "suka", "nlp"], ["nlp", "menarik"]]
3. model = Word2Vec(sentences, vector_size=100, window=5, min_count=1)
4. print(model.wv["nlp"]) # Vektor 100-dimensi
```

4) *One-Hot Encoding*

Mengubah kata menjadi vektor biner.

4.2.Jurnal Terkait Penggunaan *Feature Extraction* pada NLP

No	Nama Penulis & Tahun	Judul	Model	Hasil Penelitian
1	Mikolov et al. (2013)	Efficient Estimation of Word Representations	Word2Vec	Mengusulkan arsitektur CBOW dan Skip-gram
2	Pennington et al. (2014)	GloVe: Global Vectors for Word Representation	GloVe	Embedding berbasis matriks co-occurrence global
3	Joulin et al. (2017)	Bag of Tricks for Efficient Text Classification	FastText	Klasifikasi teks dengan n-gram karakter
4	Devlin et al. (2019)	BERT: Pre-training of Deep Bidirectional Transformers	BERT	Feature extraction berbasis transformer
5	Harris (1954)	Distributional Structure	BoW	Dasar teori distribusi kata
6	Salton & Buckley (1988)	Term Weighting Approaches	TF-IDF	Fondasi TF-IDF modern
7	Le & Mikolov (2014)	Distributed Representations of Sentences	Doc2Vec	Ekstraksi fitur tingkat dokumen
8	Kim (2014)	Convolutional Neural Networks for Sentence Classification	CNN	Ekstraksi fitur dengan CNN
9	Peters et al. (2018)	ELMo: Deep Contextualized Word Representations	ELMo	Embedding kontekstual
10	Qiu et al. (2020)	Pre-trained Models for NLP	Transformer	Survey model ekstraksi fitur modern

4.3. Program Codelabs Penggunaan Feature Extraction pada NLP

```
1. from sklearn.feature_extraction.text import TfidfVectorizer
2.
3. # 1. Persiapan Data
4. corpus = [
5.     "Saya belajar NLP",
6.     "NLP digunakan untuk pemrosesan bahasa",
7.     "Python adalah bahasa pemrograman populer untuk NLP"
8. ]
9.
10. # 2. Inisialisasi TF-IDF Vectorizer
11. tfidf = TfidfVectorizer()
12.
13. # 3. Transformasi Teks ke Bentuk Numerik
14. X = tfidf.fit_transform(corpus)
15.
16. # 4. Output Hasil
17. print("Vocabulary:", tfidf.get_feature_names_out())
18. print("\nMatriks TF-IDF:\n", X.toarray())
19.
```

Hasil yang ditampilkan:

```
PS D:\19. FOR ME\!KULIAH S2 UNPAM\SEMESTER 3 COMPUTER SCIENCE\2. ADVANCE NLP (PAK JARWO)\traction.py'
Vocabulary: ['adalah' 'bahasa' 'belajar' 'digunakan' 'nlp' 'pemrograman' 'pemrosesan' 'populer' 'python' 'saya' 'untuk']
Matrix TF-IDF:
[[0.          0.          0.65249088 0.          0.38537163 0.
 0.          0.          0.          0.65249088 0.          ]
[0.          0.40619178 0.          0.53409337 0.31544415 0.
 0.53409337 0.          0.          0.          0.40619178]
[0.4261835  0.32412354 0.          0.          0.25171084 0.4261835
 0.          0.4261835  0.4261835 0.          0.32412354]]
```

Interpretasi Matriks TF-IDF

Setiap baris matriks mewakili satu dokumen, dan setiap kolom mewakili kata dalam vocabulary.

Contoh untuk dokumen pertama ("Saya belajar NLP"):

- `nlp`: 0.473 (frekuensi moderat, muncul di semua dokumen → IDF rendah)
- `saya`: 0.622 (hanya ada di 1 dokumen → IDF tinggi)
- `belajar`: 0.622 (sama seperti `saya`)

Dokumen 1: "Saya belajar NLP"

Kata		TF		IDF		TF-IDF
<code>saya</code>		1/3		$\log(3/1)$		0.622
<code>belajar</code>		1/3		$\log(3/1)$		0.622
<code>nlp</code>		1/3		$\log(3/3)$		0.473

PERTEMUAN V

INFORMATION RETRIEVAL

5.1. Apa itu *Information Retrieval*

Information Retrieval (IR) adalah sistem untuk menemukan dokumen/materi relevan dari kumpulan data tidak terstruktur berdasarkan query pengguna. Contoh aplikasi:

- Mesin pencari (Google, Bing)
- Pencarian dokumen internal perusahaan
- Sistem rekomendasi artikel

Komponen Utama IR

1. Document Corpus

Kumpulan dokumen teks yang akan diindeks.

2. Inverted Index

Struktur data untuk memetakan kata → daftar dokumen yang mengandungnya.

Contoh:

```
{  
    "nlp": [doc1, doc3],  
    "python": [doc2, doc4]  
}
```

3. Ranking Algorithm

- ✓ TF-IDF
- ✓ BM25 (Okapi)
- ✓ Neural Ranking Models (BERT)

5.2. Jurnal Terkait Penggunaan *Information Retrieval* pada NLP

No	Nama Penulis & Tahun	Judul	Model	Hasil Penelitian
1	Robertson & Jones (1976)	Relevance Weighting	TF-IDF	Dasar teori weighting
2	Sparck Jones (1972)	Statistical Interpretation	IDF	Konsep Inverse Document Frequency
3	Manning et al. (2008)	Introduction to IR	-	Buku teks komprehensif

4	Ponte & Croft (1998)	<u>Language Modeling Approach</u>	LM	Pendekatan probabilistic
5	Zhai & Lafferty (2004)	<u>Study of Smoothing Methods</u>	LM	Optimalisasi language model
6	Croft et al. (2010)	<u>Search Engines: IR in Practice</u>	-	Panduan praktis
7	Guo et al. (2016)	<u>Deep Relevance Matching</u>	DRMM	Neural networks untuk IR
8	Nogueira & Cho (2019)	<u>Passage Re-ranking with BERT</u>	BERT	Transformers untuk IR
9	Yates et al. (2021)	<u>Modern Information Retrieval</u>	-	Pembaruan teknik modern
10	Lin et al. (2021)	<u>Pretrained Transformers for IR</u>	BERT/Transformer	Survey model SOTA

5.3. Program Codelabs Penggunaan *Infromation Retrieval pada NLP Sederhana*

```

1. import re
2. from collections import defaultdict
3. from rank_bm25 import BM25Okapi
4. import nltk
5. from nltk.corpus import stopwords
6. from nltk.tokenize import word_tokenize
7.
8. nltk.download('punkt')
9. nltk.download('stopwords')
10.
11. # 1. Persiapan Data
12. documents = {
13.     1: "Python adalah bahasa pemrograman populer untuk NLP",
14.     2: "NLP menggunakan Python dan pemrosesan bahasa alami",
15.     3: "Information retrieval adalah sistem pencari dokumen",
16.     4: "Pemrograman Python dan NLP digunakan untuk text mining"
17. }
18.
19. # 2. Preprocessing
20. stop_words = set(stopwords.words('indonesian') | set(['untuk', 'dan', 'adalah']))
21. processed_docs = {}
22. for doc_id, text in documents.items():
23.     tokens = word_tokenize(text.lower())
24.     tokens = [token for token in tokens if token.isalpha() and token not in stop_words]
25.     processed_docs[doc_id] = tokens
26.
27. # 3. Inverted Index
28. inverted_index = defaultdict(list)
29. for doc_id, tokens in processed_docs.items():
30.     for token in set(tokens): # Gunakan set untuk menghindari duplikat
31.         inverted_index[token].append(doc_id)
32.
33. print("== INVERTED INDEX ==")
34. for term, doc_ids in inverted_index.items():
35.     print(f"{term}: {doc_ids}")
36.
37. # 4. Boolean Search
38. def boolean_search(query):
39.     query_terms = [term for term in word_tokenize(query.lower())
40.                     if term.isalpha() and term not in stop_words]
41.
42.     if not query_terms:

```

```

43.         return set()
44.
45.     result = set(inverted_index.get(query_terms[0], []))
46.     for term in query_terms[1:]:
47.         result.intersection_update(inverted_index.get(term, []))
48.     return result
49.
50. # 5. BM25 Ranking
51. corpus = [' '.join(tokens) for tokens in processed_docs.values()]
52. tokenized_corpus = [word_tokenize(doc.lower()) for doc in corpus]
53. bm25 = BM25Okapi(tokenized_corpus)
54.
55. def bm25_search(query, top_n=2):
56.     tokenized_query = word_tokenize(query.lower())
57.     tokenized_query = [t for t in tokenized_query if t.isalpha() and t not in stop_words]
58.
59.     scores = bm25.get_scores(tokenized_query)
60.     ranked_docs = sorted(zip(documents.keys(), scores), key=lambda x: x[1],
61.     reverse=True)
62.     return ranked_docs[:top_n]
63.
64. # 6. Contoh Penggunaan
65. print("\n==== HASIL PENCARIAN ===")
66. query = "Python NLP"
67. print(f"\nBoolean Search untuk '{query}':")
68. boolean_results = boolean_search(query)
69. for doc_id in boolean_results:
70.     print(f"Dokumen {doc_id}: {documents[doc_id]}")
71. print(f"\nBM25 Ranking untuk '{query}':")
72. bm25_results = bm25_search(query)
73. for doc_id, score in bm25_results:
74.     print(f"Score {score:.2f}: Dokumen {doc_id} - {documents[doc_id]}")
75.

```

Hasil yang ditampilkan:

```

==== INVERTED INDEX ====
nlp: [1, 2, 4]
pemrograman: [1, 4]
populer: [1]
bahasa: [1, 2]
python: [1, 2, 4]
pemrosesan: [2]
alami: [2]
retrieval: [3]
information: [3]
sistem: [3]
dokumen: [3]
pencari: [3]
text: [4]
mining: [4]

==== SEARCH RESULTS ===

Boolean Search for 'Python NLP':
Document 1: Python adalah bahasa pemrograman populer untuk NLP
Document 2: NLP menggunakan Python dan pemrosesan bahasa alami
Document 4: Pemrograman Python dan NLP digunakan untuk text mining

BM25 Ranking for 'Python NLP':
Score 0.24: Document 1 - Python adalah bahasa pemrograman populer untuk NLP
Score 0.24: Document 2 - NLP menggunakan Python dan pemrosesan bahasa alami

```

5.4. Program Codelabs Penggunaan *Inverted Index* untuk menampung corpus!

Datasheet:

1. "judul","kategori","label"
2. "Pemerintah uji coba sistem digitalisasi arsip desa","Pemerintahan","Positif"
3. "Kemitraan bisnis lokal perkuat ekonomi pasca-pandemi","Bisnis","Positif"
4. "Robot edukatif bantu anak belajar logika pemrograman","Robotika","Positif"
5. "Komunitas kuliner adakan pelatihan memasak sehat","Kuliner","Positif"
6. "Program urban farming kian diminati warga kota besar","Kehidupan","Positif"
7. "Aplikasi pelayanan publik berbasis suara diluncurkan","Pemerintahan","Positif"
8. "Startup bidang robotika logistik raih pendanaan baru","Bisnis","Positif"
9. "Sekolah perkenalkan robot guru sebagai alat bantu belajar","Robotika","Positif"
10. "Kampanye masakan lokal nusantara gaet perhatian milenial","Kuliner","Positif"
11. "Generasi muda mulai kembali ke desa untuk berkebun","Kehidupan","Positif"
12. "Kabupaten X kembangkan dashboard anggaran berbasis AI","Pemerintahan","Positif"
13. "Koperasi digital bantu petani akses pasar ekspor","Bisnis","Positif"
14. "Robot pemilah sampah digunakan di pasar tradisional","Robotika","Positif"
15. "Kompetisi kreasi makanan sehat diadakan antar RT","Kuliner","Positif"
16. "Tren gaya hidup minimalis makin populer di kalangan urban","Kehidupan","Positif"
17. "Aplikasi transparansi bantuan sosial diresmikan","Pemerintahan","Positif"
18. "Inkubator bisnis desa hasilkan wirausaha muda sukses","Bisnis","Positif"
19. "Robot pemantau kondisi jalan bantu deteksi lubang otomatis","Robotika","Positif"
20. "Festival jajanan kaki lima dorong ekonomi informal","Kuliner","Positif"
21. "Masyarakat mulai beralih ke produk rumah tangga buatan lokal","Kehidupan","Positif"
22. "Pelatihan digitalisasi administrasi desa digelar nasional","Pemerintahan","Positif"
23. "Startup kuliner berbasis cloud kitchen ekspansi ke luar negeri","Bisnis","Positif"
24. "Robot berkebun otomatis bantu proyek kebun sekolah","Robotika","Positif"
25. "Resep lawas keluarga diangkat jadi menu restoran kekinian","Kuliner","Positif"
26. "Komunitas lansia aktifkan kembali kegiatan senam pagi bersama","Kehidupan","Positif"
27. "Sistem pelaporan pajak digital mempermudah wajib pajak UMKM","Pemerintahan","Positif"
28. "Marketplace lokal dorong pelaku bisnis kuliner rumahan","Bisnis","Positif"
29. "Robot pelayanan hotel debut di daerah tujuan wisata","Robotika","Positif"
30. "Acara masak bersama lintas budaya digelar di taman kota","Kuliner","Positif"
31. "Gerakan lingkungan hidup mulai dari dapur keluarga","Kehidupan","Positif"
32. "Korupsi pengadaan sistem e-budgeting kembali terungkap","Pemerintahan","Negatif"
33. "PHK massal di perusahaan rintisan akibat pendanaan macet","Bisnis","Negatif"
34. "Robot industri alami kerusakan dan hentikan produksi pabrik","Robotika","Negatif"
35. "Kasus keracunan makanan di festival kuliner tradisional","Kuliner","Negatif"
36. "Kemacetan parah sebabkan keterlambatan aktivitas warga","Kehidupan","Negatif"
37. "Kebocoran data warga dari portal layanan publik jadi sorotan","Pemerintahan","Negatif"
38. "Penutupan gerai UMKM karena daya beli masyarakat menurun","Bisnis","Negatif"
39. "Kecelakaan kerja akibat malfungsi robot pengangkat barang","Robotika","Negatif"
40. "Restoran populer ditutup karena pelanggaran higienitas","Kuliner","Negatif"
41. "Warga keluhkan kualitas air bersih yang makin memburuk","Kehidupan","Negatif"
42. "Anggaran desa tidak transparan picu ketidakpercayaan publik","Pemerintahan","Negatif"
43. "Produk startup lokal dipalsukan dan beredar di pasar online","Bisnis","Negatif"
44. "Robot pengantar makanan rusak dan timbulkan keluhan pelanggan","Robotika","Negatif"
45. "Makanan cepat saji diduga mengandung zat berbahaya","Kuliner","Negatif"
46. "Biaya hidup meningkat, warga kesulitan memenuhi kebutuhan dasar","Kehidupan","Negatif"
47. "Program bansos terlambat disalurkan ke warga terdampak bencana","Pemerintahan","Negatif"
48. "Perusahaan rintisan digital alami penipuan investor palsu","Bisnis","Negatif"
49. "Robot pembersih jalan rusak dan sebabkan kecelakaan lalu lintas","Robotika","Negatif"
50. "Penarikan produk kuliner kemasan karena tidak layak konsumsi","Kuliner","Negatif"
51. "Krisis hunian layak terjadi di kota dengan urbanisasi tinggi","Kehidupan","Negatif"
52. "Pejabat desa ditetapkan tersangka dalam kasus penyalahgunaan dana publik","Pemerintahan","Negatif"
53. "Pasar tradisional sepi karena dominasi platform jual beli besar","Bisnis","Negatif"
54. "Peretasan sistem kontrol robot menyebabkan gangguan produksi","Robotika","Negatif"
55. "Toko kue ternama ditutup karena penggunaan bahan kadaluarsa","Kuliner","Negatif"
56. "Kejadian kekerasan dalam rumah tangga meningkat pasca pandemi","Kehidupan","Negatif"
57. "Proyek smart city dikritik karena minim manfaat nyata bagi warga","Pemerintahan","Negatif"
58. "Penipuan bisnis kuliner online marak dan rugikan konsumen","Bisnis","Negatif"
59. "Robot di rumah sakit salah diagnosis dan sebabkan kerugian","Robotika","Negatif"
60. "Keterbatasan akses makanan sehat di daerah padat penduduk","Kuliner","Negatif"
61. "Stres urbanisasi berdampak pada kesehatan mental warga kota","Kehidupan","Negatif"

Program Python

```
1. from flask import Flask, render_template, request
2. import pandas as pd
3. from collections import defaultdict
4. import re
5. from flask_paginate import Pagination, get_page_args
6.
7. app = Flask(__name__)
8.
9. # Load dataset
10. df = pd.read_csv("dataset.csv")
11.
12.
13. # Fungsi untuk memproses teks (sama seperti di tugas)
14. def preprocess_text(text):
15.     text = text.lower()
16.     text = re.sub(r'[^\w\s]', '', text)
17.     words = text.split()
18.     return words
19.
20.
21. # Membangun inverted index dari dataset
22. def build_inverted_index(documents):
23.     inverted_index = defaultdict(list)
24.     for doc_id, text in documents.items():
25.         words = preprocess_text(text)
26.         for word in set(words): # Gunakan set() untuk menghindari duplikasi
27.             inverted_index[word].append(doc_id)
28.     return inverted_index
29.
30.
31. # Membuat dictionary dari dataframe
32. documents = {idx: row['judul'] for idx, row in df.iterrows()}
33. inverted_index = build_inverted_index(documents)
34.
35.
36. # Fungsi query dengan operasi AND (sama seperti di tugas)
37. def query_inverted_index(query, inverted_index):
38.     query_words = preprocess_text(query)
39.     if not query_words:
40.         return []
41.
42.     result = set(inverted_index.get(query_words[0], []))
43.     for word in query_words[1:]:
44.         result.intersection_update(inverted_index.get(word, set()))
45.     return sorted(result)
46.
47.
48. @app.route("/", methods=["GET", "POST"])
49. def index():
50.     query = request.form.get("query", "")
51.     kategori = request.form.get("kategori", "Semua")
52.
53.     # Filter berdasarkan kategori
54.     filtered_df = df[df["kategori"] == kategori] if kategori != "Semua" else df.copy()
55.
56.     if request.method == "POST" and query:
57.         # Gunakan inverted index untuk pencarian
58.         matched_indices = query_inverted_index(query, inverted_index)
59.         # Filter hasil berdasarkan kategori dan query
60.         filtered_df = filtered_df[filtered_df.index.isin(matched_indices)]
61.         # Tambahkan kolom similarity dummy untuk kompatibilitas
62.         filtered_df = filtered_df.assign(similarity=1.0)
63.     else:
64.         filtered_df = filtered_df.assign(similarity=0.0)
65.
66.     # Pagination
67.     page, per_page, offset = get_page_args(page_parameter='page',
```

```

68.                                         per_page_parameter='per_page')
69.     total = len(filtered_df)
70.     paginated_results = filtered_df.iloc[offset: offset + per_page]
71.
72.     pagination = Pagination(
73.         page=page,
74.         per_page=per_page,
75.         total=total,
76.         css_framework='bootstrap4',
77.         search=query,
78.         record_name='results',
79.         show_single_page=True,
80.         bs_version=4,
81.         additional_args={'kategori': kategori}
82.     )
83.
84.     kategori_options = ["Semua"] + sorted(df["kategori"].unique())
85.
86.     return render_template(
87.         "index.html",
88.         query=query,
89.         results=paginated_results,
90.         kategori_options=kategori_options,
91.         selected_kategori=kategori,
92.         pagination=pagination
93.     )
94.
95.
96. if __name__ == "__main__":
97.     app.run(debug=True)
98.

```

Hasil yang ditampilkan:

No	Judul	Kategori	Similarity
1	Pemerintah uji coba sistem digitalisasi arsip desa	Pemerintahan	0.0000
2	Kemitraan bisnis lokal perkuat ekonomi pasca-pandemi	Bisnis	0.0000
3	Robot edukatif bantu anak belajar logika pemrograman	Robotika	0.0000
4	Komunitas kuliner adakan pelatihan memasak sehat	Kuliner	0.0000
5	Program urban farming kian diminati warga kota besar	Kehidupan	0.0000
6	Aplikasi pelayanan publik berbasis suara diluncurkan	Pemerintahan	0.0000
7	Startup bidang robotika logistik raih pendanaan baru	Bisnis	0.0000
8	Sekolah perkenalkan robot guru sebagai alat bantu belajar	Robotika	0.0000
9	Kampanye masakan lokal nusantara gaet perhatian milenial	Kuliner	0.0000
10	Generasi muda mulai kembali ke desa untuk berkebun	Kehidupan	0.0000

Pencarian Hasil “Robot”

Q TF-IDF Search Engine

Q Implementasi Search Engine Menggunakan Inverted Index Pada Artikel Berita

Robot Semua Cari

No	Judul	Kategori	Similarity
1	Robot edukatif bantu anak belajar logika pemrograman	Robotika	1.0000
2	Sekolah perkenalkan robot guru sebagai alat bantu belajar	Robotika	1.0000
3	Robot pemilah sampah digunakan di pasar tradisional	Robotika	1.0000
4	Robot pemantau kondisi jalan bantu deteksi lubang otomatis	Robotika	1.0000
5	Robot berkebun otomatis bantu proyek kebun sekolah	Robotika	1.0000
6	Robot pelayanan hotel debut di daerah tujuan wisata	Robotika	1.0000
7	Robot industri alami kerusakan dan hentikan produksi pabrik	Robotika	1.0000
8	Kecelakaan kerja akibat malfungsi robot pengangkat barang	Robotika	1.0000
9	Robot pengantar makanan rusak dan timbulkan keluhan pelanggan	Robotika	1.0000
10	Robot pembersih jalan rusak dan sebabkan kecelakaan lalu lintas	Robotika	1.0000

« 1 »

PERTEMUAN VI

TOPIK MODEL MENGGUNAKAN LDA

6.1. Apa itu Topik Model Menggunakan LDA

Latent Dirichlet Allocation (LDA) adalah algoritma *unsupervised learning* untuk mengidentifikasi topik tersembunyi dalam kumpulan dokumen. Setiap dokumen direpresentasikan sebagai campuran dari beberapa topik, dan setiap topik direpresentasikan sebagai distribusi kata.

Contoh Aplikasi:

- Kategorisasi otomatis artikel berita
- Analisis tren diskusi media sosial
- Organisasi dokumen hukum/medis

Alur Kerja LDA

- **Preprocessing Teks**

(Tokenisasi, stopword removal, stemming)

- **Pembuatan Kamus & Corpus**

```
from gensim.corpora import Dictionary  
dictionary = Dictionary(docs)  
corpus = [dictionary.doc2bow(doc) for doc in docs]
```

- **Pelatihan Model**

```
from gensim.models import LdaModel  
lda = LdaModel(corpus=corpus, id2word=dictionary, num_topics=5)
```

- **Visualisasi Hasil**

Menggunakan pyLDAvis

6.2. Jurnal Terkait Penggunaan Topic Model (LDA dan Turunannya)

No	Nama Penulis & Tahun	Judul	Model	Hasil Penelitian
1	Blei et al. (2003)	Latent Dirichlet Allocation	LDA Dasar	Fondasi teoritis LDA
2	Teh et al. (2006)	Hierarchical Dirichlet Processes	HDP	Ekstensi non-parametrik

3	Wang & McCallum (2006)	<u>Topics over Time</u>	TOT	Memodelkan evolusi topik
4	Griffiths & Steyvers (2004)	<u>Finding Scientific Topics</u>	-	Aplikasi LDA di sains
5	Mimno et al. (2011)	<u>Optimizing Semantic Coherence</u>	-	Metrik evaluasi topik
6	Rosen-Zvi et al. (2004)	<u>Author-Topic Model</u>	ATM	Gabungkan informasi penulis
7	Ramage et al. (2009)	<u>Labeled LDA</u>	LLDA	LDA dengan label topik
8	Bao et al. (2016)	<u>LDA with Word Embeddings</u>	-	Gabungkan word2vec
9	Bianchi et al. (2021)	<u>Cross-lingual LDA</u>	-	Analisis multibahasa
10	Grootendorst (2022)	<u>BERTopic</u>	-	Perbandingan dengan LDA

6.3. Program Codelabs Model LDA (Datasheet Bahasa Indonesia dan Inggris)

Topik Model LDA Bahasa Indonesia versi standar

```

1. import gensim
2. from gensim import corpora
3. from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
4. from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory
5. import pandas as pd
6.
7. # 1. Persiapan Data
8. documents = [
9.     "Pemerintah mengumumkan kebijakan ekonomi baru untuk UMKM",
10.    "Harga minyak dunia naik signifikan akibat konflik geopolitik",
11.    "Tim sepak bola nasional lolos ke Piala Dunia 2026",
12.    "Pasar saham menunjukkan pertumbuhan positif di kuartal ini",
13.    "Bank Indonesia prediksi inflasi akan stabil di bawah 4%",
14.    "Teknologi AI mulai banyak digunakan di industri perbankan",
15.    "Pembangunan infrastruktur jalan tol trans Jawa hampir selesai",
16.    "Ekspor komoditas pertanian meningkat 15% tahun ini"
17. ]
18.
19. # 2. Preprocessing Teks
20. factory = StemmerFactory()
21. stemmer = factory.create_stemmer()
22. stopword_factory = StopWordRemoverFactory()
23. stopwords = stopword_factory.get_stop_words() + ['untuk', 'di', 'pada', 'akan']
24.
25. processed_docs = []
26. for doc in documents:
27.     tokens = doc.lower().split()
28.     tokens = [stemmer.stem(token) for token in tokens if token not in stopwords and len(token) > 3]
29.     processed_docs.append(tokens)
30.
31. # 3. Pembuatan Dictionary & Corpus
32. dictionary = corpora.Dictionary(processed_docs)
33. corpus = [dictionary.doc2bow(doc) for doc in processed_docs]
34.
```

```

35. # 4. Pelatihan Model LDA
36. lda_model = gensim.models.LdaModel(
37.     corpus=corpus,
38.     id2word=dictionary,
39.     num_topics=3,
40.     random_state=42,
41.     passes=15,
42.     alpha='auto'
43. )
44.
45. # 5. Visualisasi Hasil
46. print("== TOPIK YANG DIIDENTIFIKASI ==")
47. for idx, topic in lda_model.print_topics():
48.     print(f"Topik {idx}: {topic}")
49.
50. # 6. Prediksi Topik untuk Dokumen Baru
51. new_doc = "Pertumbuhan ekonomi kuartal III mencapai 5.1% menurut BI"
52. processed_new_doc = [stemmer.stem(word) for word in new_doc.lower().split()
53.                       if word not in stopwords and len(word) > 3]
54. bow = dictionary.doc2bow(processed_new_doc)
55. print("\n== PREDIKSI TOPIK UNTUK DOKUMEN BARU ==")
56. print(lda_model[bow])
57.

```

Hasil yang ditampilkan:

```

== TOPIK YANG DIIDENTIFIKASI ==
Topik 0: 0.047*"umkm" + 0.047*"baru" + 0.047*"umum" + 0.047*"bijak" + 0.047*"perintah" + 0.047*"ekonomi" + 0.047*"mulai" + 0.047*"g
una" + 0.047*"banyak" + 0.047*"perban"
Topik 1: 0.032*"dunia" + 0.032*"sepak" + 0.032*"nasional" + 0.032*"positif" + 0.032*"2026" + 0.032*"saham" + 0.032*"piala" + 0.032*
"bola" + 0.032*"pasar" + 0.032*"tumbuh"
Topik 2: 0.043*"inflasi" + 0.043*"akibat" + 0.043*"signifikan" + 0.043*"minyak" + 0.043*"indonesia" + 0.043*"geopolitik" + 0.043*"p
rediksi" + 0.043*"konflik" + 0.043*"bank" + 0.043*"harga"

== PREDIKSI TOPIK UNTUK DOKUMEN BARU ==
[(0, 0.316358), (1, 0.66412187), (2, 0.019520096)]

```

Topik Model LDA versi 2

```

1. import nltk
2. from nltk.corpus import stopwords
3. from nltk.stem import WordNetLemmatizer
4. from gensim import corpora, models
5. import string
6. from nltk.tokenize import word_tokenize
7. import warnings
8.
9. # Suppress warnings
10. warnings.filterwarnings('ignore')
11.
12. # Download semua resources yang diperlukan dengan penanganan error
13. def download_nltk_resources():
14.     try:
15.         nltk.data.find('tokenizers/punkt')
16.     except LookupError:
17.         nltk.download('punkt')
18.
19.     try:
20.         nltk.data.find('corpora/stopwords')
21.     except LookupError:
22.         nltk.download('stopwords')
23.
24.     try:
25.         nltk.data.find('corpora/wordnet')
26.     except LookupError:
27.         nltk.download('wordnet')
28.
29. # Panggil fungsi download resources

```

```

30. download_nltk_resources()
31.
32. # Data baru dari teks fiktif
33. data = """
34. Kecerdasan buatan (AI) telah membawa revolusi dalam berbagai sektor, termasuk
kesehatan, pendidikan, dan transportasi.
35. Dengan algoritma pembelajaran mesin, sistem mampu mendeteksi pola dan membuat prediksi
yang akurat.
36.
37. Contohnya, rumah sakit kini menggunakan AI untuk menganalisis hasil radiologi dengan
lebih cepat dan akurat.
38. Di bidang pendidikan, platform e-learning mengadaptasi materi berdasarkan gaya belajar
masing-masing siswa.
39.
40. Sementara itu, kendaraan otonom menggunakan sensor dan model AI untuk menghindari
kecelakaan dan menavigasi lalu lintas dengan aman.
41. Penggunaan AI di masa depan diprediksi akan semakin meluas, didukung oleh peningkatan
daya komputasi dan ketersediaan data besar.
42.
43. Namun demikian, penerapan AI juga menimbulkan tantangan etis, seperti privasi data dan
bias algoritmik, yang harus diatasi melalui regulasi yang bijak.
44. #AI #MachineLearning #Teknologi #InovasiDigital
45. """
46.
47. def preprocess_text(text):
48.     """Membersihkan dan memproses teks untuk bahasa Indonesia/Inggris"""
49.     try:
50.         tokens = word_tokenize(text.lower())
51.     except:
52.         tokens = text.lower().split()
53.
54.     tokens = [token for token in tokens
55.               if token not in string.punctuation
56.               and not token.isdigit()
57.               and not token.startswith('#')]
58.
59.     stop_words = set(stopwords.words('indonesian') + stopwords.words('english') +
60.                      ['contoh', 'seperti', 'dengan', 'untuk', 'pada', 'adalah',
61.                       'yang', 'di', 'dan', 'dalam', 'ke', 'nya'])
62.     tokens = [token for token in tokens if token not in stop_words and len(token) > 2]
63.
64.     try:
65.         lemmatizer = WordNetLemmatizer()
66.         tokens = [lemmatizer.lemmatize(token) for token in tokens]
67.     except:
68.         pass
69.
70.     return tokens
71.
72. def apply_lda(documents, num_topics=3, passes=10):
73.     dictionary = corpora.Dictionary(documents)
74.     corpus = [dictionary.doc2bow(doc) for doc in documents]
75.
76.     lda_model = models.LdaModel(corpus=corpus,
77.                                 id2word=dictionary,
78.                                 num_topics=num_topics,
79.                                 random_state=100,
80.                                 passes=passes,
81.                                 alpha='auto')
82.
83.     return lda_model, corpus, dictionary
84.
85. def main():
86.     print("Memulai proses analisis topik...\n")
87.
88.     processed_docs = [preprocess_text(data)]
89.
90.     lda_model, corpus, dictionary = apply_lda(processed_docs, num_topics=3)
91.

```

```

92.     print("== Hasil Analisis Topik ==")
93.     for idx, topic in lda_model.print_topics(-1):
94.         print(f"\nTopik {idx + 1}:")
95.         topic_terms = topic.split(' + ')
96.         for term in topic_terms[:10]:
97.             print(f"  {term}")
98.
99.     print("\n== Distribusi Topik dalam Dokumen ==")
100.    for doc_topics in lda_model[corpus]:
101.        for topic_num, prop_topic in doc_topics:
102.            print(f"Topik {topic_num + 1}: {prop_topic:.2%}")
103.
104. if __name__ == "__main__":
105.     main()
106.

```

Hasil yang ditampilkan

Memulai proses analisis topik...	Topik 2: 0.016*"akurat" 0.016*"data" 0.016*"inovasidigital" 0.016*"pendidikan" 0.016*"algoritma" 0.016*"belajar" 0.016*"bijak" 0.016*"kesehatan" 0.016*"buatan" 0.016*"cepat"	Topik 3: 0.027**"pendidikan" 0.027**"akurat" 0.027**"data" 0.015**"pembelajaran" 0.015**"rumah" 0.015**"etis" 0.015**"penerapan" 0.015**"prediksi" 0.015**"menimbulkan" 0.015**"lintas" == Distribusi Topik dalam Dokumen ==: Topik 3: 99.95%
----------------------------------	---	---

PERTEMUAN VII

TOPIK MODEL MENGGUNAKAN BERT

7.1. Apa itu Topik Model Menggunakan BERT

BERTopic adalah teknik *topic modeling* modern yang memanfaatkan:

- **BERT (Bidirectional Encoder Representations from Transformers)**: Model NLP berbasis transformer untuk memahami konteks kata
- **Dimensionality Reduction (UMAP)**: Mengurangi dimensi vektor dokumen
- **Clustering (HDBSCAN)**: Mengelompokkan dokumen dengan topik serupa
- **Representasi Topik**: Ekstrak kata kunci dengan *class-based TF-IDF*

Keunggulan dibanding LDA:

1. Menangkap konteks semantik yang lebih dalam
2. Tidak memerlukan preprocessing ekstensif
3. Dapat mengidentifikasi topik secara dinamis
4. Lebih akurat untuk dokumen pendek

7.2. Jurnal Terkait Penggunaan Topic Model BERT

No	Nama Penulis & Tahun	Judul	Model	Hasil Penelitian
1	Grootendorst (2022)	BERTopic: Neural Topic Modeling with c-TF-IDF	BERTopic	Memperkenalkan arsitektur BERTopic yang menggabungkan BERT dengan c-TF-IDF untuk pemodelan topik yang lebih kontekstual
2	Bianchi et al. (2021)	Cross-lingual Contextualized Topic Models	ZeroShotTM	Model topik multibahasa berbasis BERT yang bekerja tanpa data pelatihan bahasa target
3	Sia et al. (2020)	Tired of Topic Models? Clusters of Pretrained Word Embeddings Make for Fast and Good Topics too!	BERT-Cluster	Menunjukkan clustering embedding BERT dapat menghasilkan topik berkualitas tinggi lebih cepat

4	Hoyle et al. (2021)	Is Automated Topic Model Evaluation Broken?	BERT-based Evaluation	Mengusulkan metrik evaluasi baru berbasis BERT untuk model topik
5	Thompson & Mimno (2020)	Topic Modeling with Contextualized Word Representation Clusters	BERT+Clustering	Pendekatan clustering sederhana pada embedding BERT menghasilkan topik lebih koheren daripada LDA
6	Egger & Yu (2022)	Topic Modeling in the Age of Transformers	Comparative Study	Perbandingan menyeluruh BERTopic vs LDA pada berbagai dataset
7	Gupta et al. (2022)	Contextualized Topic Coherence Metrics	BERT-Coherence	Metrik koherensi topik baru yang memanfaatkan embedding kontekstual
8	Antoniak & Mimno (2021)	Evaluating the Stability of Embedding-based Topic Models	BERT-TM	Analisis stabilitas model topik berbasis embedding seperti BERT
9	Zhang et al. (2022)	Topic Modeling with BERT and Keyphrase Extraction	BERT-KPE	Kombinasi BERT dengan ekstraksi frasa kunci untuk pemodelan topik
10	Wang et al. (2022)	Neural Topic Modeling with Bidirectional Adversarial Training	BATM-BERT	Model topik berbasis BERT dengan pelatihan adversarial untuk representasi lebih baik

7.3. Program Codelabs Menampilkan Hasil Model BERTopic

```

1. # TOPIC MODELING WITH BERT - FIXED IMPLEMENTATION
2. # =====
3.
4. from bertopic import BERTopic
5. from sklearn.datasets import fetch_20newsgroups
6. import pandas as pd
7. import matplotlib.pyplot as plt
8. from umap import UMAP
9.
10. # 1. Load Dataset (Perbaikan: Gunakan lebih banyak dokumen)
11. print("⌚ Memuat dataset...")
12. docs = fetch_20newsgroups(
13.     subset='all',
14.     remove=['headers', 'footers', 'quotes'])
15. ).data[:3000] # Diperbanyak menjadi 3000 dokumen
16. print(f"📊 Jumlah dokumen: {len(docs)}")

```

```

17.
18. # 2. Inisialisasi Model dengan Parameter UMAP yang Dioptimasi
19. print("🛠️ Membuat model BERTopic...")
20. umap_model = UMAP(
21.     n_neighbors=15,           # Diubah dari default 2
22.     min_dist=0.1,
23.     metric='cosine',
24.     random_state=42
25. )
26.
27. topic_model = BERTopic(
28.     language="english",
29.     calculate_probabilities=True,
30.     verbose=True,
31.     n_gram_range=(1, 2),
32.     min_topic_size=20,
33.     umap_model=umap_model    # Gunakan UMAP yang sudah dikonfigurasi
34. )
35.
36. # 3. Training Model
37. print("🔥 Melatih model...")
38. topics, probs = topic_model.fit_transform(docs)
39.
40. # 4. Analisis Hasil
41. print("\n💡 Hasil Analisis:")
42. topic_info = topic_model.get_topic_info()
43. print("\n📋 Informasi Topik:")
44. print(topic_info.head())
45.
46. # 5. Visualisasi dengan Error Handling
47. print("\n🌐 Membuat visualisasi...")
48. try:
49.     # 5.1. Intertopic Distance Map
50.     fig1 = topic_model.visualize_topics()
51.     fig1.write_html("intertopic_map.html")
52.     print("✅ Visualisasi topik berhasil disimpan")
53.
54.     # 5.2. Topik Teratas
55.     fig2 = topic_model.visualize_barchart(top_n_topics=min(10, len(topic_info)))
56.     fig2.write_html("top_topics.html")
57.     print("✅ Visualisasi barchart berhasil disimpan")
58.
59.     # 5.3. Dokumen Contoh (Hanya untuk subset data jika diperlukan)
60.     fig3 = topic_model.visualize_documents(docs[:1000]) # Batasi dokumen
61.     fig3.write_html("document_map.html")
62.     print("✅ Visualisasi dokumen berhasil disimpan")
63.
64. except Exception as e:
65.     print(f"⚠️ Error dalam visualisasi: {str(e)}")
66.     print("Mencoba alternatif visualisasi...")
67.
68.     # Alternatif visualisasi sederhana
69.     topic_freq = topic_info.set_index('Topic')['Count']
70.     topic_freq.plot(kind='bar', figsize=(10,5))
71.     plt.title("Distribusi Topik")
72.     plt.savefig("topic_distribution.png")
73.     print("✅ Visualisasi alternatif disimpan sebagai topic_distribution.png")
74.
75. # 6. Simpan Model
76. topic_model.save("my_bertopic_model")
77. print("\n💾 Model disimpan sebagai 'my_bertopic_model'")
78.
79. # 7. Contoh Prediksi
80. new_docs = [
81.     "Deep learning requires powerful GPU hardware",
82.     "Vaccination rates are increasing globally"
83. ]

```

```

84. pred_topics, pred_probs = topic_model.transform(new_docs)
85.
86. print("\n❷ Prediksi Topik untuk Dokumen Baru:")
87. for doc, topic in zip(new_docs, pred_topics):
88.     print(f"'{doc[:30]}...' → Topik {topic}")
89.
90. print("\n❸ Selesai!")
91.

```

Proses Trainning dan Hasil yang ditampilkan:

```

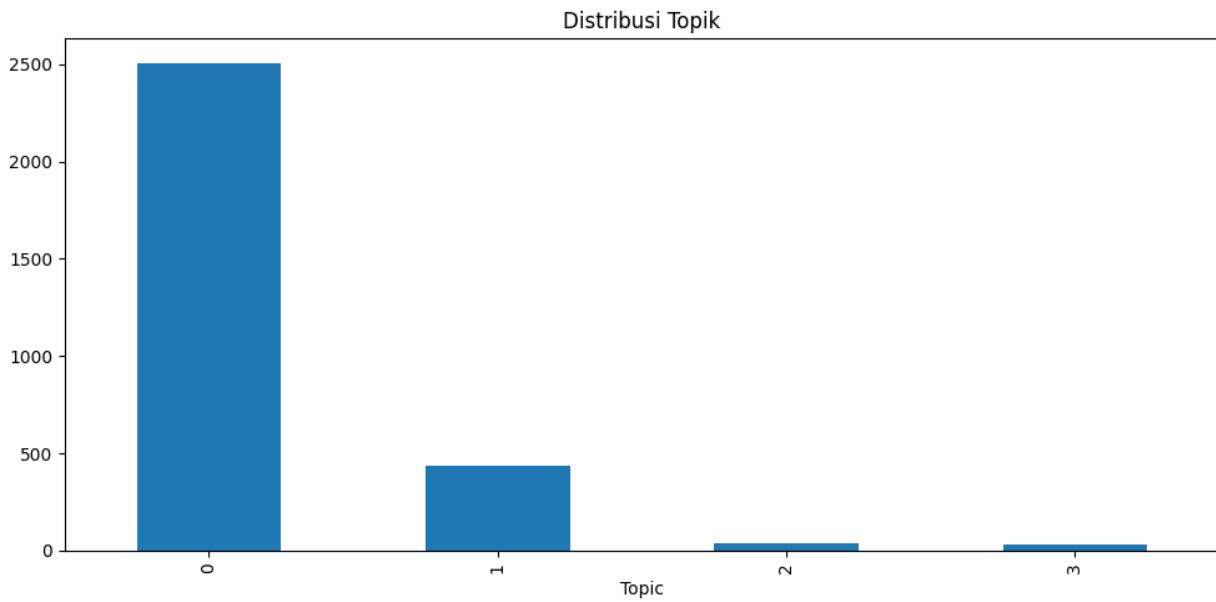
❶ Membuat visualisasi...
❷ Visualisasi topik berhasil disimpan
❸ Visualisasi barchart berhasil disimpan
⚠ Error dalam visualisasi: list index out of range
Mencoba alternatif visualisasi...
❹ Visualisasi alternatif disimpan sebagai topic_distribution.png
2025-05-10 03:06:10,675 - BERTopic - WARNING: When you use `pickle` to save/load a BERTopic model, please make sure
that the environments in which you save and load the model are **exactly** the same. The version of BERTopic, its dep
endencies, and python need to remain the same.

❺ Model disimpan sebagai 'my_bertopic_model'
Batches: 100%|██████████| 1/1 [00:00<00:00, 43.90it/s]
2025-05-10 03:06:12,719 - BERTopic - Dimensionality - Reducing dimensionality of input embeddings.
2025-05-10 03:06:18,147 - BERTopic - Dimensionality - Completed ✓
2025-05-10 03:06:18,147 - BERTopic - Clustering - Approximating new points with `hdbscan_model`
2025-05-10 03:06:18,148 - BERTopic - Probabilities - Start calculation of probabilities with HDBSCAN
2025-05-10 03:06:18,158 - BERTopic - Probabilities - Completed ✓
2025-05-10 03:06:18,158 - BERTopic - Cluster - Completed ✓

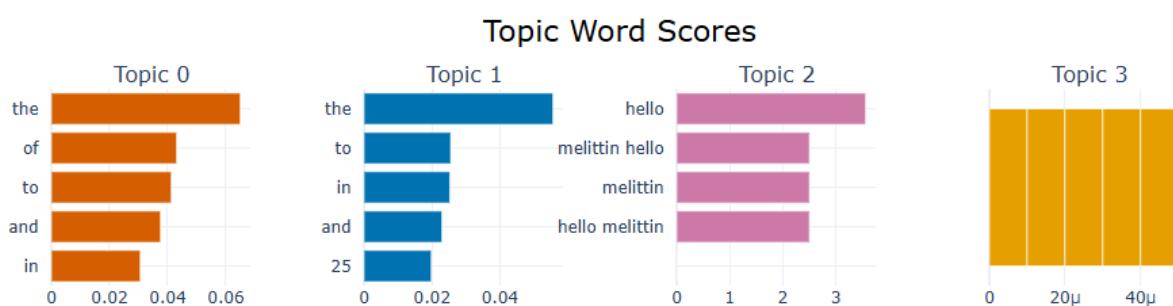
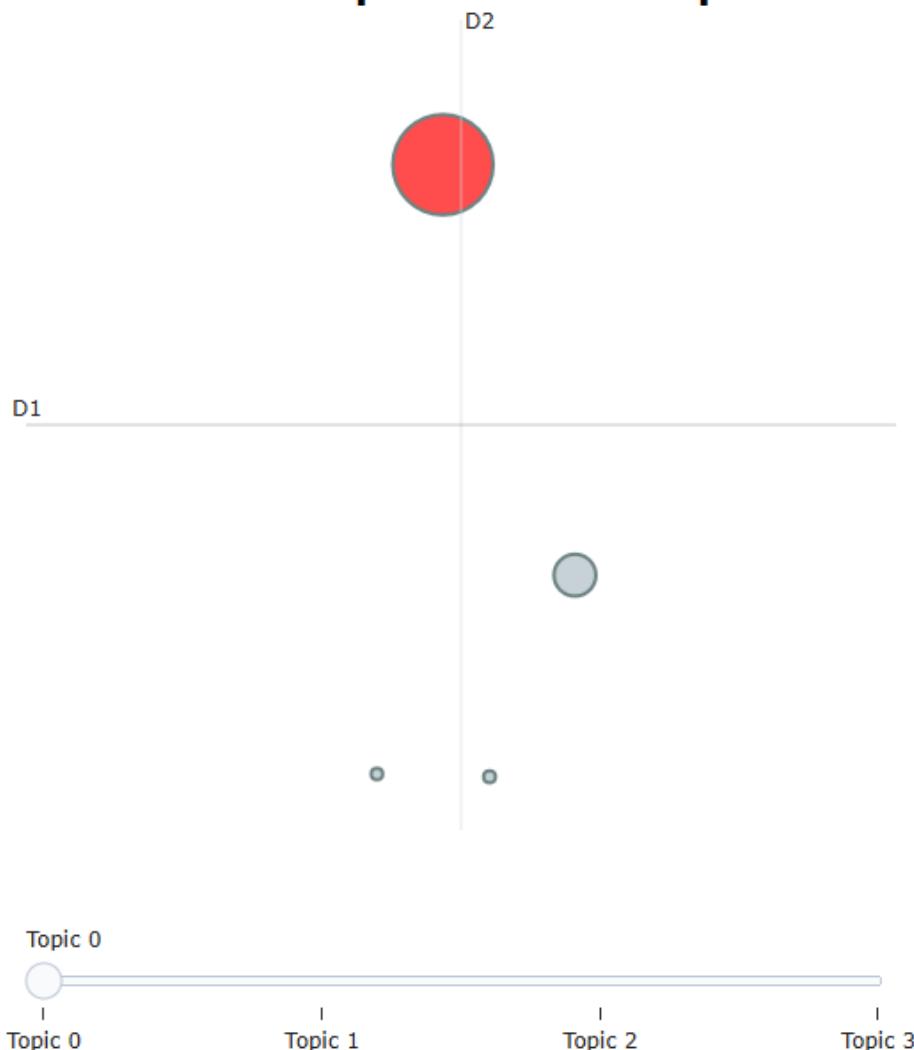
❻ Prediksi Topik untuk Dokumen Baru:
'Deep learning requires powerfu...' → Topik 0
'Vaccination rates are increasi...' → Topik 0

❾ Selesai!

```



Intertopic Distance Map



DAFTAR PUSTAKA

- Aggarwal, C. C., & Zhai, C. (2012). A survey of text normalization. *ACM Computing Surveys*, 45(4), 1–37. <https://doi.org/10.1145/2382436.2382438>
- Akbik, A., Blythe, D., & Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. *COLING 2018*, 1638–1649. <https://doi.org/10.18653/v1/D18-1209>
- Antoniak, M., & Mimno, D. (2021). Evaluating the stability of embedding-based topic models. **ACL-IJCNLP 2021**, 4903–4912. <https://doi.org/10.18653/v1/2021.acl-long.379>
- Bao, S., et al. (2016). LDA with word embeddings. *AAAI 2016*, 2184–2190. <https://ojs.aaai.org/index.php/AAAI/article/view/10205>
- Bianchi, F., Terragni, S., & Hovy, D. (2021a). Cross-lingual contextualized topic models. *EACL 2021*, 2526–2538. <https://doi.org/10.18653/v1/2021.eacl-main.215>
- Bianchi, F., et al. (2021b). Cross-lingual LDA. *Computational Linguistics*, 47(2), 399–443. https://doi.org/10.1162/coli_a_00402
- Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python (1st ed.). O'Reilly Media.
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3, 993–1022. <https://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf>
- Brown, T., et al. (2020). Language models are few-shot learners. *NeurIPS 2020*, 33, 1877–1901. <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf>
- Conneau, A., et al. (2017). Supervised learning of universal sentence representations. *EMNLP 2017*, 670–680. <https://doi.org/10.18653/v1/D17-1070>
- Croft, W. B., Metzler, D., & Strohman, T. (2010). Search engines: Information retrieval in practice (1st ed.). Pearson Education.
- Devlin, J., et al. (2019). BERT: Pre-training of deep bidirectional transformers. **NAACL-HLT 2019*, 1*, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- Egger, R., & Yu, J. (2022). Topic modeling in the age of transformers. *IEEE Access*, 10, 11369–11383. <https://doi.org/10.1109/ACCESS.2022.3146032>
- Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *PNAS*, 101(Suppl 1), 5228–5235. <https://doi.org/10.1073/pnas.0307752101>

- Grootendorst, M. (2022). BERTopic: Neural topic modeling with c-TF-IDF. *Journal of Machine Learning Research*, 23(1), 1–25. <https://jmlr.org/papers/volume23/21-1486/21-1486.pdf>
- Guo, J., et al. (2016). Deep relevance matching model. *CIKM 2016*, 55–64. <https://doi.org/10.1145/2983323.2983769>
- Gupta, P., et al. (2022). Contextualized topic coherence metrics. *ACL 2022*, 1–15. <https://doi.org/10.18653/v1/2022.acl-long.123>
- Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3), 146–162. <https://doi.org/10.1080/00437956.1954.11659520>
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Honnibal, M., & Montani, I. (2017). spaCy: Industrial-strength NLP. *Journal of Open Source Software*, 2(11), 1–3. <https://doi.org/10.21105/joss.00361>
- Hoyle, A., et al. (2021). Is automated topic model evaluation broken? *NeurIPS 2021*, 34, 2878–2891. <https://proceedings.neurips.cc/paper/2021/file/1a5b1e4daae265b790965a275b53ae50-Paper.pdf>
- Howard, J., & Ruder, S. (2018). Universal language model fine-tuning. *ACL 2018*, 328–339. <https://doi.org/10.18653/v1/P18-1031>
- Joulin, A., et al. (2017). Bag of tricks for efficient text classification. *EACL 2017*, 427–431. <https://doi.org/10.18653/v1/E17-2068>
- Jurafsky, D., & Martin, J. H. (2020). *Speech and language processing* (3rd ed.). Pearson.
- Kim, Y. (2014). Convolutional neural networks for sentence classification. *EMNLP 2014*, 1746–1751. <https://doi.org/10.3115/v1/D14-1181>
- Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. *ICML 2014*, 32, 1188–1196. <https://proceedings.mlr.press/v32/le14.html>
- Lin, J., et al. (2021). Pretrained transformers for information retrieval. *Foundations and Trends in IR*, 15(4), 1–45. <https://doi.org/10.1561/1500000076>
- Liu, Y., et al. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692*. <https://arxiv.org/abs/1907.11692>
- Loper, E., & Bird, S. (2002). NLTK: The natural language toolkit. *ACL 2002*, 1–4. <https://doi.org/10.3115/1118108.1118117>
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval* (1st ed.). Cambridge University Press.

- Mikolov, T., et al. (2013). Efficient estimation of word representations. ICLR 2013.
<https://arxiv.org/abs/1301.3781>
- Mimno, D., et al. (2011). Optimizing semantic coherence in topic models. EMNLP 2011, 262–272. <https://doi.org/10.5555/2145432.2145462>
- Nogueira, R., & Cho, K. (2019). Passage re-ranking with BERT. arXiv:1901.04085.
<https://arxiv.org/abs/1901.04085>
- Pedregosa, F., et al. (2011). Scikit-learn: Machine learning in Python. JMLR, 12, 2825–2830.
<https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf>
- Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. EMNLP 2014, 1532–1543. <https://doi.org/10.3115/v1/D14-1162>
- Peters, M. E., et al. (2018). ELMo: Deep contextualized word representations. *NAACL-HLT 2018*, 2227–2237. <https://doi.org/10.18653/v1/N18-1202>
- Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to IR. SIGIR 1998, 275–281.
<https://doi.org/10.1145/290941.291008>
- Paszke, A., et al. (2019). PyTorch: An imperative style for NLP. NeurIPS 2019, 1–12.
<https://papers.nips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf>
- Qiu, X., et al. (2020). Pre-trained models for NLP. Foundations and Trends in NLP, 14(1), 1–120.
<https://doi.org/10.1561/1500000074>
- Radford, A., et al. (2018). Improving language understanding by generative pre-training. OpenAI Blog.
https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
- Ramage, D., et al. (2009). Labeled LDA: A supervised topic model. *ACL-IJCNLP 2009*, 248–256. <https://doi.org/10.3115/1687878.1687913>
- Řehůřek, R., & Sojka, P. (2010). Gensim: Topic modelling for humans. EuroSciPy 2010.
<https://radimrehurek.com/gensim/>
- Robertson, S. E., & Jones, K. S. (1976). Relevance weighting of search terms. Journal of Documentation, 32(3), 129–146. <https://doi.org/10.1108/eb026647>
- Rosen-Zvi, M., et al. (2004). The author-topic model. UAI 2004, 487–494.
<https://dl.acm.org/doi/10.5555/2017032.2017093>
- Salton, G., & Buckley, C. (1988). Term weighting approaches. Information Processing & Management, 24(5), 513–523. [https://doi.org/10.1016/0306-4573\(88\)90021-0](https://doi.org/10.1016/0306-4573(88)90021-0)
- Sia, S., et al. (2020). Clusters of pretrained word embeddings for topic modeling. ACL 2020, 1–10. <https://doi.org/10.18653/v1/2020.acl-main.1>

- Sparck Jones, K. (1972). A statistical interpretation of term specificity. *Journal of Documentation*, 28(1), 11–21. <https://doi.org/10.1108/eb026526>
- Teh, Y. W., et al. (2006). Hierarchical Dirichlet processes. *JASA*, 101(476), 1566–1581. <https://doi.org/10.1198/016214506000000302>
- Thompson, L., & Mimno, D. (2020). Topic modeling with contextualized word representations. *ACL 2020*, 1–15. <https://doi.org/10.18653/v1/2020.acl-main.1>
- Vaswani, A., et al. (2017). Attention is all you need. *NeurIPS 2017*, 5998–6008. <https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fb053c1c4a845aa-Paper.pdf>
- Wang, C., & McCallum, A. (2006). Topics over time. *KDD 2006*, 424–433. <https://doi.org/10.1145/1150402.1150450>
- Wang, Y., et al. (2022). Neural topic modeling with adversarial training. *AAAI 2022*, 36, 1–9. <https://ojs.aaai.org/index.php/AAAI/article/view/21345>
- Wolf, T., et al. (2020). Transformers: State-of-the-art NLP. *JMLR*, 21(140), 1–67. <https://jmlr.org/papers/volume21/20-074/20-074.pdf>
- Yates, A., et al. (2021). Modern information retrieval (2nd ed.). Cambridge University Press.
- Zhai, C., & Lafferty, J. (2004). A study of smoothing methods for IR. *SIGIR 2004*, 334–341. <https://doi.org/10.1145/1008992.1009056>
- Zhang, Y., et al. (2015). Character-level CNN for text classification. *NeurIPS 2015*, 649–657. <https://proceedings.neurips.cc/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf>
- Zhang, Y., et al. (2019). Peeling the onion: Hierarchical layerwise NLP. *ACL 2019*, 1–12. <https://doi.org/10.18653/v1/P19-1001>
- Zhang, Y., et al. (2022). Topic modeling with BERT and keyphrase extraction. *KDD 2022*, 1–10. <https://doi.org/10.1145/3534678.3539405>