Text Similarity Techniques: TF-IDF vs. BERT

Author: Sherin Johny
Student ID: 23050521
Github: Github Repository

1.Introduction

In the realm of natural language processing (NLP), text similarity is a crucial concept that enables machines to understand the relationships between different text documents or queries. This capability is fundamental to various AI applications, including search engines, recommendation systems, and question-answering platforms.

This tutorial explores two key techniques for text representation TF-IDF and BERT and demonstrates how they can be combined with cosine similarity to create basic AI assistants. These assistants can identify the most relevant responses to user questions by comparing the similarity between the query and a pre-existing knowledge base.

2.Understanding Text Representation

Before delving into similarity measurements, it's essential to understand how we convert human language into numerical formats that computers can process. We'll examine two powerful approaches for this conversion:

2.1 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a statistical method that evaluates the importance of words within a document collection. It operates on the principle that words unique to a specific document but rare across all documents are likely the most informative

2.1.1 How TF-IDF Works:

Architecture of TF-IDF

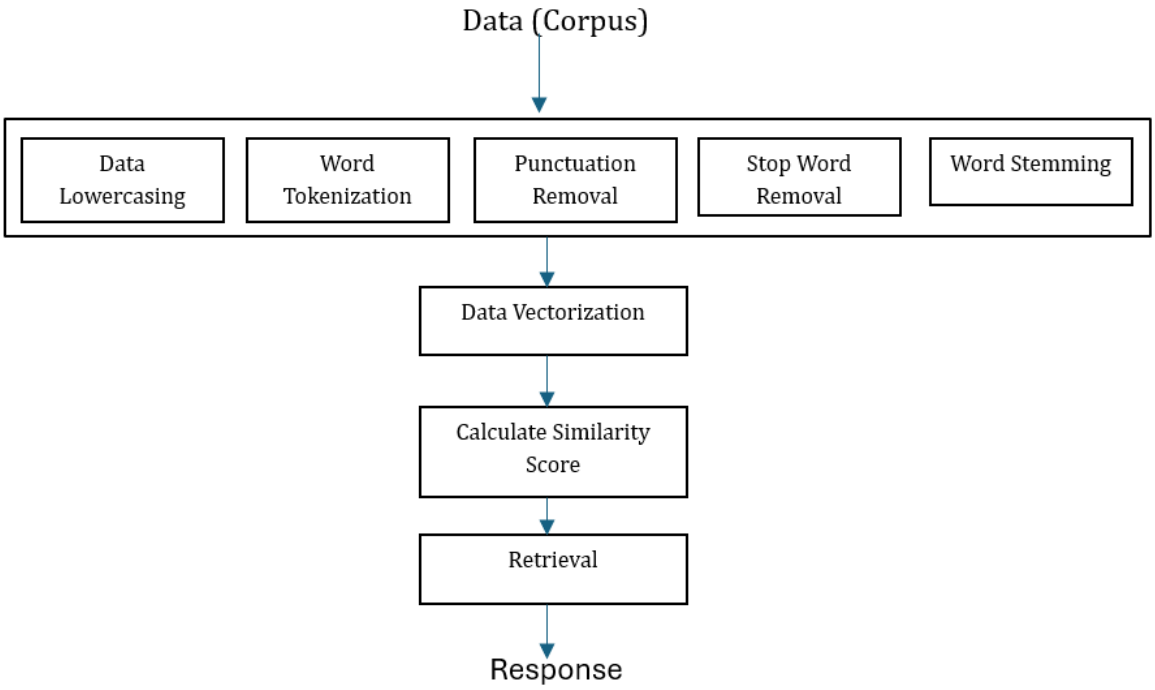

Figure 1. Architecture of TF-IDF: "Diagram illustrating the TF-IDF process, showing text input, term frequency calculation, inverse document frequency calculation, and final TF-IDF score output. The diagram depicts the flow from raw text to a weighted numerical representation of terms." Image reference

TF-IDF combines two metrics:

1.Term Frequency (TF): Measures how frequently a term appears in a document.

$$\text{TF}(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

2.Inverse Document Frequency (IDF): Measures how important a term is by considering how rare it is across all documents.

$$\text{IDF}(t, D) = \log\left(\frac{\text{Total number of documents in corpus D}}{\text{Number of documents containing term t}}\right)$$

TF-IDF Score is the product of these two metrics.

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D)$$

2.1.2 Key Characteristics of TF-IDF:

- **Bag-of-Words Approach**: Disregards grammar and word order
- **Sparse Representation**: Generates high-dimensional vectors with many zero values
- **Computationally Efficient**: Requires minimal resources
- **Language-Agnostic**: Applicable across different languages without modification
- **Limited Semantic Understanding**: Cannot capture context or word relationships

For instance, in a document about "Python programming," the word "Python" would have a high TF-IDF score because it appears frequently in that document but not in documents about other topics. Conversely, common words like "the" or "and" would have low scores due to their frequent appearance across all documents.

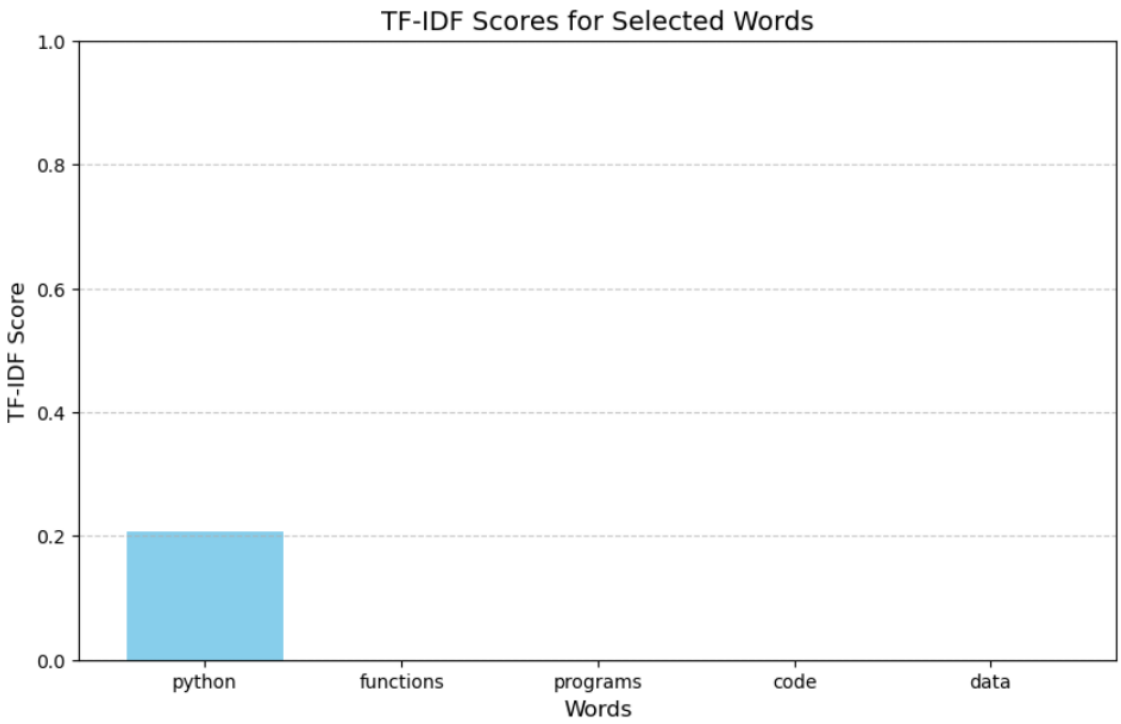

Figure 2. TF-IDF Score for Words: "Bar chart showing the TF-IDF scores for different words in a sample document. The x-axis lists the words, and the y-axis represents the TF-IDF score. The chart illustrates that words like 'Python' have high TF-IDF scores in a document about Python programming, while common words have low scores." (Python Code for the plot)

2.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT signifies a substantial advancement in NLP, transitioning from basic statistical methods to neural network-based approaches that capture deeper semantic meaning. Developed by Google in 2018, BERT is pre-trained on extensive text data and can generate context-aware representations of words and sentences.

2.2.1 How BERT Works:

BERT processes text through several steps:

1. Tokenization: Breaks text into tokens (words or subwords)
2. Embedding: Converts tokens into initial vector representations
3. Contextual Processing: Uses transformer architecture to consider surrounding context
4. Output Embeddings: Produces context-aware embeddings for each token
5. Sentence Representation: Often uses the [CLS] token embedding to represent the entire sentence

Architecture of BERT

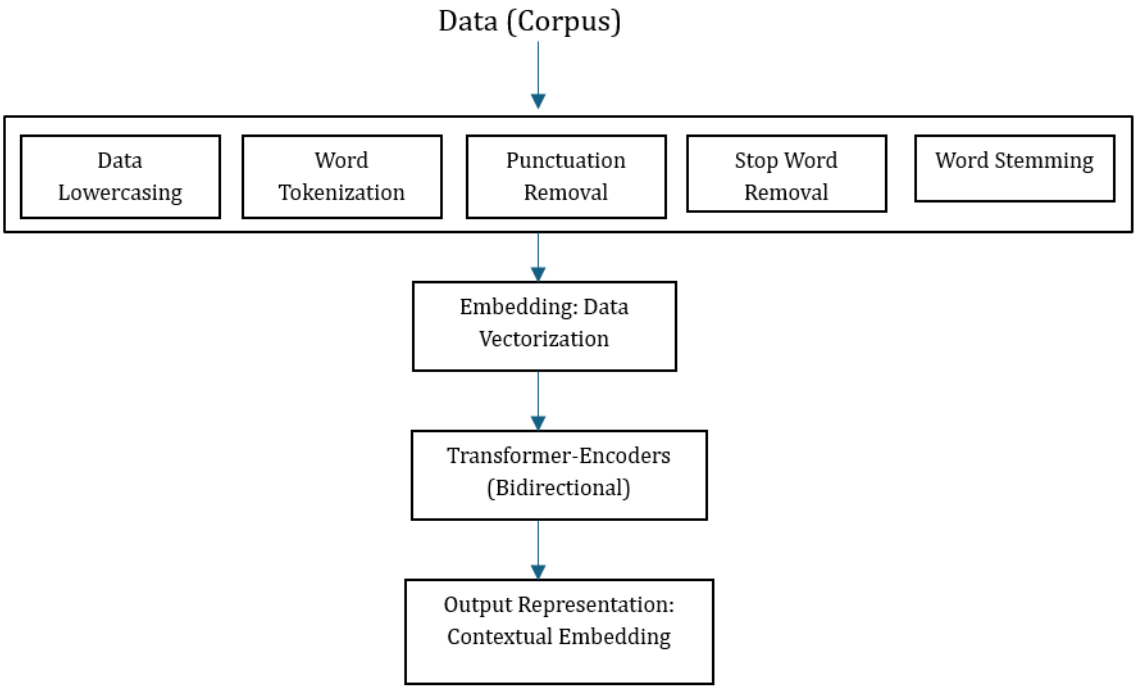

Figure 3. BERT Architecture: "Diagram of the BERT (Bidirectional Encoder Representations from Transformers) architecture, highlighting the input embeddings, transformer layers, and output embeddings. The diagram shows how BERT processes text bidirectionally to capture context-aware representations." Image reference

2.2.2 Key Characteristics of BERT:

- **Contextual Understanding**: Represents words differently based on their context
- **Dense Representation**: Creates fixed-length, dense vectors
- **Pre-trained Knowledge**: Captures semantic relationships from massive pre-training
- **Resource-Intensive**: Requires significant computational resources
- **Strong Semantic Understanding**: Effectively captures nuanced meanings and relationships

For example, BERT can understand that in "I'm going to the bank to deposit money" and "I'm sitting by the riverbank," the word "bank" has completely different meanings based on context—something TF-IDF would miss entirely.

3. Measuring Similarity with Cosine Similarity

Once we have vector representations of text, we need a way to measure their similarity. Cosine similarity is a popular metric for this purpose

3.1 Understanding Cosine Similarity

Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space. The resulting value ranges from -1 (completely opposite) to 1 (exactly the same), with 0 indicating orthogonality (no similarity).

The mathematical formula for cosine similarity is:

$$\text{Cosine Similarity}(A, B) = \frac{(A \cdot B)}{(||A|| \times ||B||)}$$

Where:

- $A \cdot B$ is the dot product of vectors A and B
- $||A||$ is the magnitude (or Euclidean norm) of vector A
- $||B||$ is the magnitude of vector B

Why Cosine Similarity Works Well for Text:

- Focuses on orientation rather than magnitude
- Performs well in high-dimensional spaces

- Provides easily interpretable results (0 to 1 range)
- Is computationally efficient, especially for sparse vectors

For example, if we have two documents with similar topics but different lengths, cosine similarity will still show a high similarity score because it measures the angle between the vectors, not their lengths.

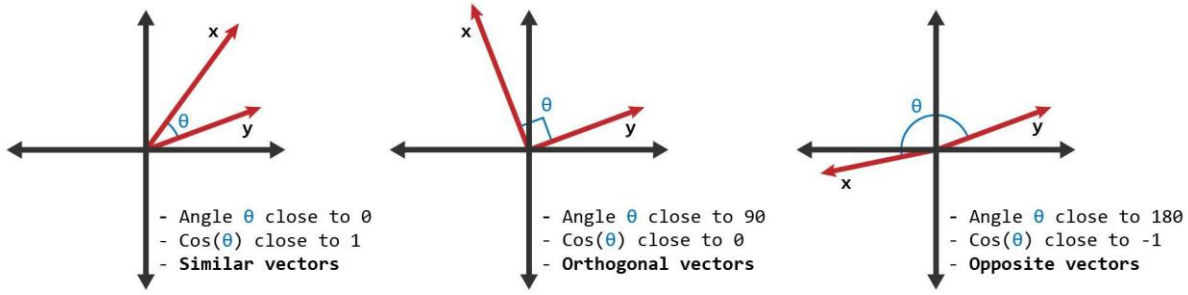

Figure 4. Cosine similarity measures the cosine of the angle between two vectors. A smaller angle indicates higher similarity: "Illustration of cosine similarity, showing two vectors and the angle between them. The diagram explains that a smaller angle indicates higher similarity between the vectors, with angles closer to 0 degrees representing high similarity and angles closer to 90 degrees representing low similarity." Image Source

4. Building AI Assistants with Text Similarity

Let's explore how these techniques can be combined to create basic AI assistants that answer user questions by finding the most similar questions in a knowledge base.

A link to the Github Repository of the Jupiter Notebook with the implementation of TF-IDF Assistant and BERT Assistant is attached here. Packages to be installed are included.

Please download the notebook, execute and compare the performance of the AI assistants.

Link to the Jupiter Notebook: **TF_IDF_BERT_code.ipynb**

4.1 TF-IDF Based Assistant

Steps in building the assistant:

1. Prepare a knowledge base of question-answer pairs
2. Convert all questions to TF-IDF vectors
3. Process user queries by converting them to TF-IDF vectors
4. Calculate cosine similarity between the query vector and knowledge base vectors
5. Return the answer associated with the most similar question

The below figure shows how TF-IDF works for a Query Answering system.

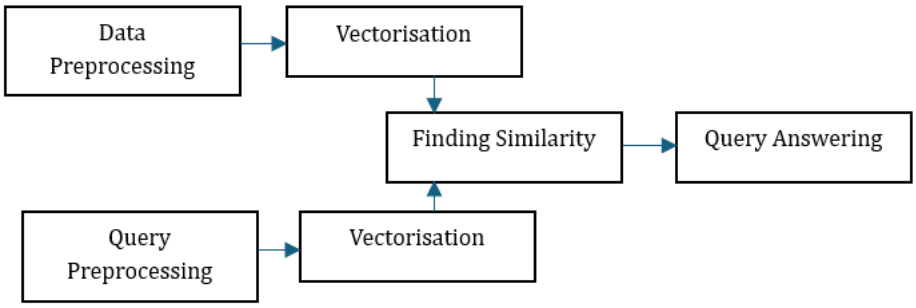

Figure 5. TF-IDF for a Query Answering system: "Flowchart depicting a TF-IDF-based query answering system. The flowchart shows the process of converting both the knowledge base questions and the user query into TF-IDF vectors, calculating cosine similarity, and retrieving the answer associated with the most similar question." Image Reference

Advantages of TF-IDF Assistants:

- Simple to implement and understand
- Efficient processing with minimal computational requirements
- No training required
- Transparent relationship between keywords and matching

Limitations of TF-IDF Assistants:

- Heavily reliant on exact keyword matches
- Cannot handle synonyms or paraphrasing
- Ignores word order and context
- May require language-specific tuning for stop words and stemming

4.2 BERT-Based Assistant

A BERT-based assistant follows a similar approach but uses BERT embeddings:

1. Prepare the knowledge base
2. Generate BERT embeddings for all questions
3. Create BERT embeddings for user queries
4. Calculate cosine similarity between query and knowledge base embeddings
5. Return the most similar answer

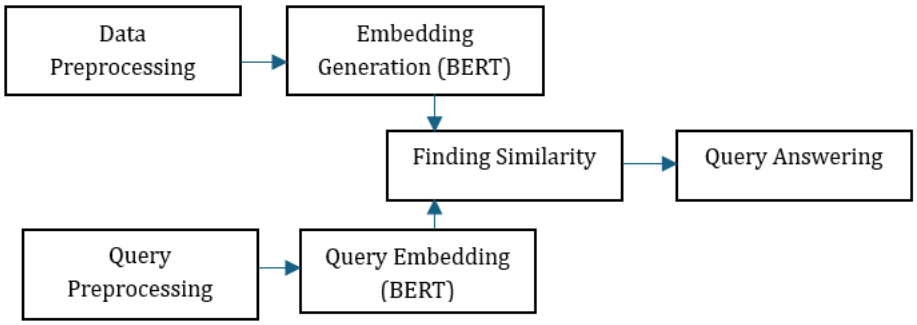

Figure 6. BERT for a Query Answering system: "Flowchart illustrating a BERT-based query answering system. The flowchart shows the process of converting both the knowledge base questions and the user query into BERT embeddings, calculating cosine similarity, and retrieving the answer associated with the most similar question." Image Reference

Advantages of BERT Assistants:

- Captures meaning beyond simple keyword matching
- Considers the full context of words and phrases
- Can match questions phrased differently but with the same meaning
- Benefits from pre-training knowledge

Limitations of BERT Assistants:

- Higher computational cost
- More complex to implement
- Less transparent about matching reasons
- Limited by maximum sequence length (typically 512 tokens)

4.3 Comparing TF-IDF and BERT Approaches

When deciding which approach to use for your AI assistant, consider these key differences:

Aspect	**TF-IDF with Cosine Similarity**	**BERT with Cosine Similarity**
Complexity	Simple and easy to implement.	Advanced and requires more resources.
Semantic Understanding	No semantic understanding	Captures semantic meaning and context.
Performance	Works well for simple queries.	Handles complex and ambiguous queries.

Computational Cost	Low (can run on a standard laptop).	High (requires GPUs for large datasets).
Use Case	Small to medium-sized corpora	Large corpora and complex tasks.
Implementation	Minimal setup and dependencies.	Requires pre-trained models and tools

Table 1. "Table comparing TF-IDF with Cosine Similarity and BERT with Cosine Similarity across key aspects: Complexity, Semantic Understanding, Performance, Computational Cost, Use Case, and Implementation. TF-IDF is simpler, faster, and suitable for small to medium datasets but lacks semantic understanding. BERT is more advanced, resource-intensive, and better for large datasets, offering deeper contextual analysis."

4.4 Practical Considerations and Enhancements

To improve your AI assistant, consider these enhancements:

- **Ensemble Methods**: Combine results from both TF-IDF and BERT
- **Two-Stage Processing**: Use TF-IDF for initial filtering, then BERT for refined ranking
- **Feature Augmentation**: Combine TF-IDF features with BERT embeddings
- **Fine-tuning**: Adapt BERT to your specific domain or task

5.Conclusion

Text similarity techniques are foundational to many AI assistants, with TF-IDF and BERT representing different generations of approaches. While TF-IDF offers simplicity and efficiency, BERT provides deeper semantic understanding at the cost of increased computational requirements.

For practical applications:

- Use TF-IDF when resources are limited, response time is critical, or your knowledge base contains distinctive keywords.
- Use BERT when semantic understanding is crucial, users might phrase questions variably, or accuracy is more important than speed.

As you build and refine your AI assistants, remember that the choice between TF-IDF and BERT (or newer models) depends on your specific use case, resources, and requirements. By understanding these foundational techniques, you'll be well-equipped to create effective AI assistants that can understand and respond to user queries across a wide range of applications.

REFERENCES:

1. DEVLIN, J., CHANG, M. W., LEE, K., & TOUTANOVA, K. (2018). "BERT: PRE-TRAINING OF DEEP BIDIRECTIONAL TRANSFORMERS FOR LANGUAGE UNDERSTANDING." ARXIV PREPRINT ARXIV:1810.04805.
2. ROBERTSON, S. E., & JONES, K. S. (1976). RELEVANCE WEIGHTING OF SEARCH TERMS. JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, 27(3), 129-146.
3. SALTON, G., & BUCKLEY, C. (1988). TERM-WEIGHTING APPROACHES IN AUTOMATIC TEXT RETRIEVAL. INFORMATION PROCESSING & MANAGEMENT, 24(5), 513-523.
4. ALAMMAR, J. (2018). "THE ILLUSTRATED BERT, ELMO, AND CO." BLOG POST.
    a. URL: HTTPS://JALAMMAR.GITHUB.IO/ILLUSTRATED-BERT/
5. SCIKIT-LEARN DOCUMENTATION. "TEXT FEATURE EXTRACTION"
    a. URL: HTTPS://SCIKIT-LEARN.ORG/STABLE/MODULES/FEATURE_EXTRACTION.HTML
6. HUGGING FACE TRANSFORMERS DOCUMENTATION
    a. URL: HTTPS://HUGGINGFACE.CO/DOCS/TRANSFORMERS/MODEL_DOC/BERT
7. GOOGLE AI BLOG. "OPEN-SOURCING BERT: STATE-OF-THE-ART PRE-TRAINING FOR NATURAL LANGUAGE PROCESSING"
    a. URL: HTTPS://AI.GOOGLEBLOG.COM/2018/11/OPEN-SOURCING-BERT-STATE-OF-ART-PRE.HTML