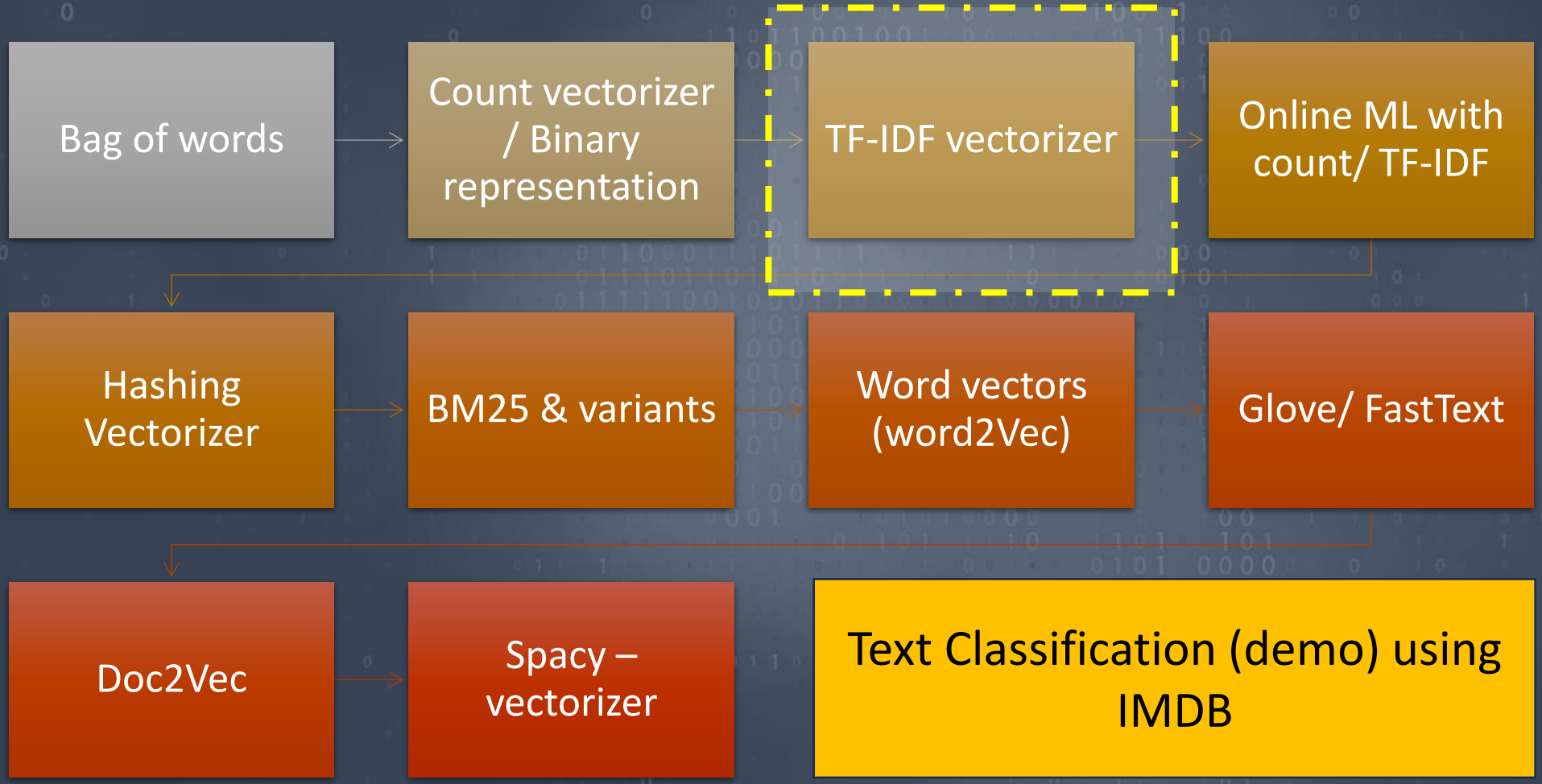
Vectorization & Embedding

"Textual Alchemy: Words to numbers"

GROKKERS
AI FOR EVERYONE

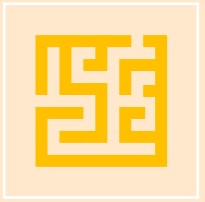
Topics to cover



Vectorizer – TF-IDF

"TF-IDF: Your Semantic Bridge"

how high counts in count vect poses problem



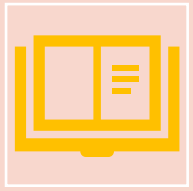
High counts in count vectorization, while seemingly informative, can pose several problems

very high word counts can dominate the representation and introduce noise

Overemphasis on Common Words

Words that appear frequently across many documents (often known as "stop words" like "the," "and," "is") can have extremely high counts in multiple documents.

In count vectorization, these common words can dominate the representation.



Overemphasizing common words can lead to less discriminative power in the vectors, as these words may not carry meaningful information about the content of a document.

Lack of Discrimination:

When a word occurs many times within a single document, it may dominate the representation of that document.

This can be problematic because a high count of a specific word within a document may not necessarily indicate its importance or relevance.

Discriminating between documents based on such high counts may not capture the nuances and subtleties of their content.

Noise in the Data

Extremely high word counts can introduce noise into the representations.

Noise can affect the performance of NLP models, especially when using simple counting-based representations like Bag of Words (BoW).

Loss of Context

Count vectorization, including BoW, discards information about word order and word semantics.

When a word is counted excessively, its context within the document may be lost, potentially impacting the quality of the representation.

mitigate the problems associated with high counts

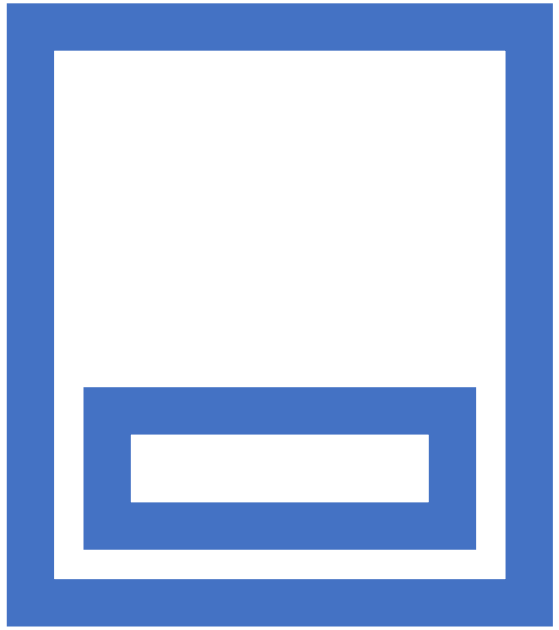
- **Term Frequency-Inverse Document Frequency (TF-IDF):** TF-IDF is one way to address the dominance of common words by assigning lower weights to them based on their inverse document frequency. This reduces the impact of high counts for common words.

- **Text Preprocessing:** Techniques like stop word removal and stemming or lemmatization can help reduce the frequency of common words and variations of words, improving the quality of vector representations.

TF-IDF (Term Frequency-Inverse Document Frequency)

- is a numerical statistic used in natural language processing (NLP) to evaluate the importance of a word in a document relative to its frequency in the entire corpus of documents.

- is widely used in various NLP tasks, including information retrieval, text classification, and document ranking.



Term Frequency (TF)

The first part of TF-IDF, the Term Frequency (TF), measures how frequently a term (word) appears in a document.

It quantifies how often a word occurs within the document.

The TF of a term in a document is calculated using …

TF(t, d) = (Number of times term t appears in document d) / (Total number of terms in document d)

Inverse Document Frequency (IDF):

- The second part of TF-IDF, the Inverse Document Frequency (IDF), measures the importance of a term across a collection of documents (corpus).

- It quantifies how common or rare a word is in the entire corpus.

- The IDF of a term is calculated using the following formula:

IDF(t) = log((Total number of documents in the corpus) / (Number of documents containing term t))

TF-IDF Calculation:

- To calculate the TF-IDF score for a term (word) in a document, you multiply its Term Frequency (TF) in that document by its Inverse Document Frequency (IDF) across the entire corpus.

- The TF-IDF score for a term t in a document d is calculated as follows:

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

Example

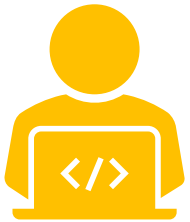
- Consider a corpus with three documents:
  - Document 1: "The quick brown fox jumps over the lazy dog."
  - Document 2: "A brown cat plays with a dog."
  - Document 3: "The lazy cat sleeps.“
- Let's calculate the TF-IDF score for the term "dog" in Document 1:
  - TF(dog, Document 1) = 1 (since "dog" appears once in Document 1)
  - IDF(dog) = log(3 / 2) ≈ 0.176 (since "dog" appears in two out of three documents)
  - TF-IDF(dog, Document 1) = TF(dog, Document 1) * IDF(dog) = 1 * 0.176 ≈ 0.176

Interpretation

- A higher TF-IDF score indicates that a term is both frequent within a specific document (high TF) and relatively rare across the entire corpus (high IDF).

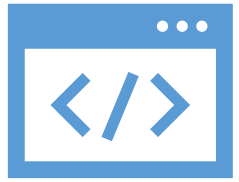
- Terms with higher TF-IDF scores are considered more important and indicative of the content of a document.

Use cases of TF-IDF



Information Retrieval: TF-IDF was originally developed for information retrieval systems.

Document Summarization: High TF-IDF values can indicate words that are important in a document.



Keyword Extraction: TF-IDF can be used to extract keywords from a document

Other use cases

Document Clustering and Topic Modeling: In unsupervised machine learning, TF-IDF vectors of documents can serve as input to clustering algorithms (like K-means) or topic modeling methods (like LDA) to group documents by their content.

Text Classification: TF-IDF can transform raw texts into feature vectors which can then be used in machine learning models. For instance, in sentiment analysis, spam detection, or categorizing articles into predefined topics.

Recommender Systems: TF-IDF can be used in content-based recommendation systems. For example, by comparing the TF-IDF vectors of different articles, a system can recommend articles that are semantically similar to a given article.

Textual Similarity: The cosine similarity between TF-IDF vectors can provide a measure of the similarity between two documents. This is used in applications like plagiarism detection.

Advantages of TF-IDF

Measures Word Importance:

- TF-IDF captures the importance of a word in a document by considering both its frequency in the document (Term Frequency) and its rarity across the entire document collection (Inverse Document Frequency).

Handles Common Words:

- TF-IDF down-weights common words (stop words) that may appear frequently across all documents but are not informative.

Discriminates Against Frequent Terms:

- While Term Frequency (TF) measures how often a term appears in a document, TF-IDF penalizes terms that are too frequent, giving more weight to terms that are relatively rare in the document collection.

Limitations

Ignores Word Order and Semantics:

TF-IDF treats documents as bags of words, ignoring word order and semantic relationships between words.

Limited Context Understanding:

TF-IDF focuses on individual terms and does not capture the context in which terms appear.

Fixed Vocabulary Size:

The vocabulary used in TF-IDF is often fixed based on the training data. New words in test or unseen data won't be represented unless the vocabulary is updated, potentially leading to information loss.

… some more limitations

Equal Weighting of Terms:

In the TF component of TF-IDF, all terms are given equal weight. This can be problematic when certain terms are more important than others but have similar frequencies.

Insensitive to Synonyms and Variants:

Synonyms or different word forms (e.g., plurals) are treated as separate terms by TF-IDF.

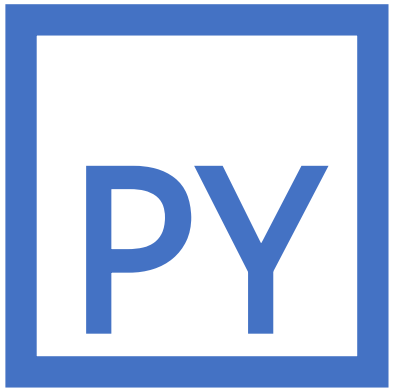
Not Effective for Short Documents:

TF-IDF may not perform well on very short documents where the frequency of terms is low. In such cases, the importance of terms may be exaggerated, and the results may be less reliable.

Sparse Representation:

The TF-IDF matrix can be sparse, especially when dealing with a large vocabulary or sparse text data. This can lead to increased memory requirements and computational inefficiency.

Demo using python/sklearn



(Implement text classification using TF-IDF vectorizer)

Qs

- What is the primary distinction between count vectorization and TF-IDF vectorization?
  - **A.** Count vectorization gives a binary representation, whereas TF-IDF provides a frequency-based representation.
  - **B.** Count vectorization is based on term frequency only, while TF-IDF also considers inverse document frequency.
  - **C.** Count vectorization can't handle bigrams or trigrams, while TF-IDF can.
  - **D.** Count vectorization is used exclusively for text, while TF-IDF is used for images.
- **Answer:** B

Qs

- Which of the following statements best describes the relevance of TF-IDF to modern architectures like BERT and GPT?

  - A. BERT and GPT internally use TF-IDF for token embedding.
  - B. BERT and GPT replace the need for TF-IDF with attention mechanisms.
  - C. TF-IDF is the primary mechanism behind the success of models like BERT and GPT.
  - D. BERT was trained using TF-IDF as its primary loss function.

- Answer: B

Qs

- In traditional NLP applications, why might one choose TF-IDF over simple count vectorization?

  - A. TF-IDF inherently understands semantic meanings between words.
  - B. TF-IDF amplifies the weight of rare words across documents.
  - C. TF-IDF has a better computational efficiency than count vectorization.
  - D. TF-IDF exclusively works with neural networks, while count vectorization does not.

- Answer: B

Qs

- Consider a corpus with a document containing the sentence "The sun shines bright." Using count vectorization, what will be the representation for the word "sun"?
    - A. The frequency of the word "sun" in the entire corpus.
    - B. The binary representation indicating the presence of the word "sun."
    - C. The inverse document frequency of the word "sun."
    - D. The semantic meaning associated with the word "sun."
- Answer: A

Qs

- Why might TF-IDF not be as essential for Transformer architectures like BERT as it is for older models?

  - A. Transformer architectures don't process text sequentially.
  - B. Transformer architectures internally implement TF-IDF in their attention mechanisms.
  - C. The attention mechanisms in Transformer architectures can dynamically weigh the importance of tokens based on context.
  - D. Transformer architectures only use pre-trained embeddings, eliminating the need for TF-IDF.

- Answer: C

Qs

- How does the attention mechanism in architectures like GPT and BERT compare to TF-IDF?

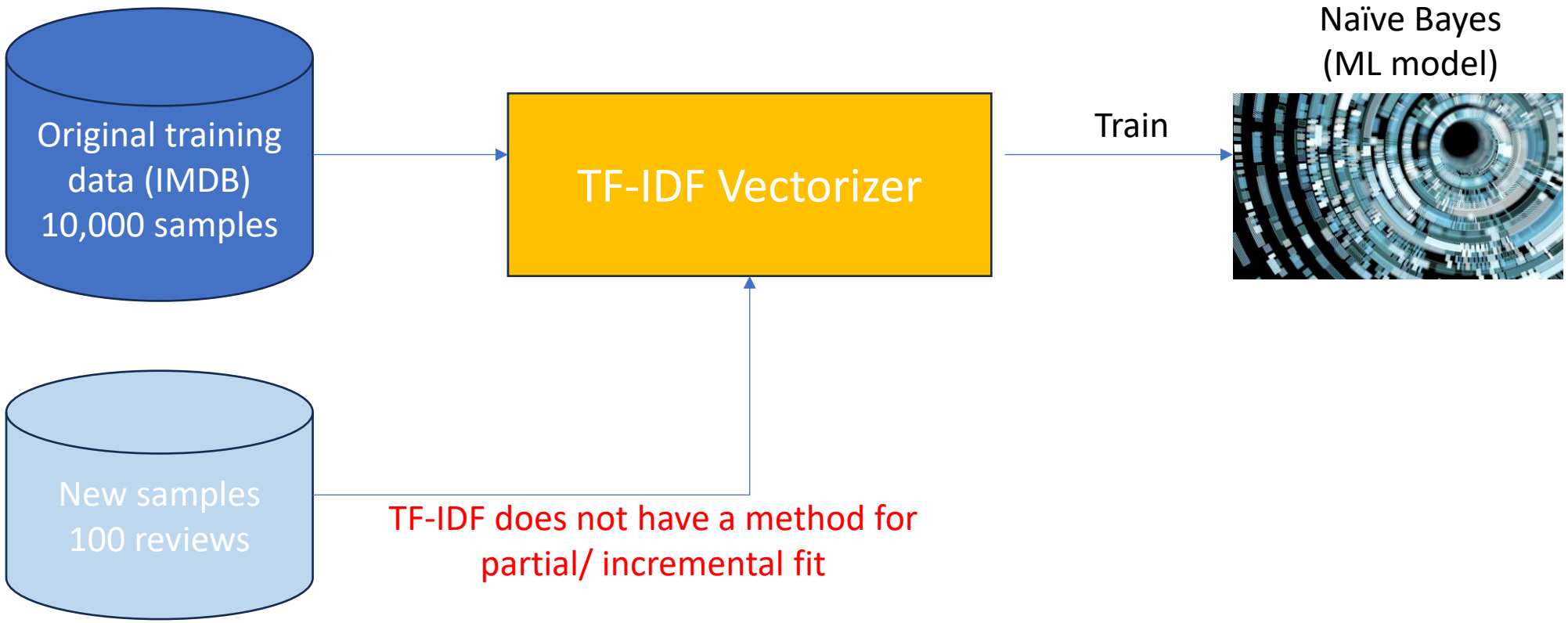
  - A. Both provide a static weight to tokens based on their importance in a corpus.
  - B. Attention mechanisms dynamically assign weights based on token relationships in context, while TF-IDF gives static weights based on term and document frequencies.
  - C. Attention mechanisms replace all traditional NLP tokenization and weighting techniques.
  - D. Both are based on the frequency of token occurrence, without considering the context.

- Answer: B

Count & TF-IDF vectorizer

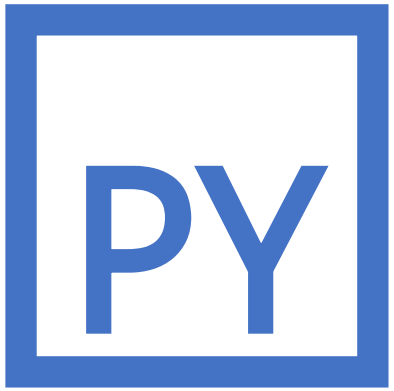
for online learning (ML)

Online ML model



Demo using python/sklearn

(Implement text classification using TF-IDF vectorizer for online ML)



Thanks!