Document Similarity Project

Presented by Nour Yaakoub

Core Concepts

Document Similarity is used to measure the similarity between two pieces of text using custom implementations of tokenization, vectorization, and similarity calculation.

01. Text Preprocessing

02. TF-IDF

03. Cosine Similarity

Workflow Overview

01
Input Texts

02
Preprocessing

03
Vocabulary Building

04
Mapping

05
TF and IDF Calculation

06
Vectorization

07
Cosine Similarity

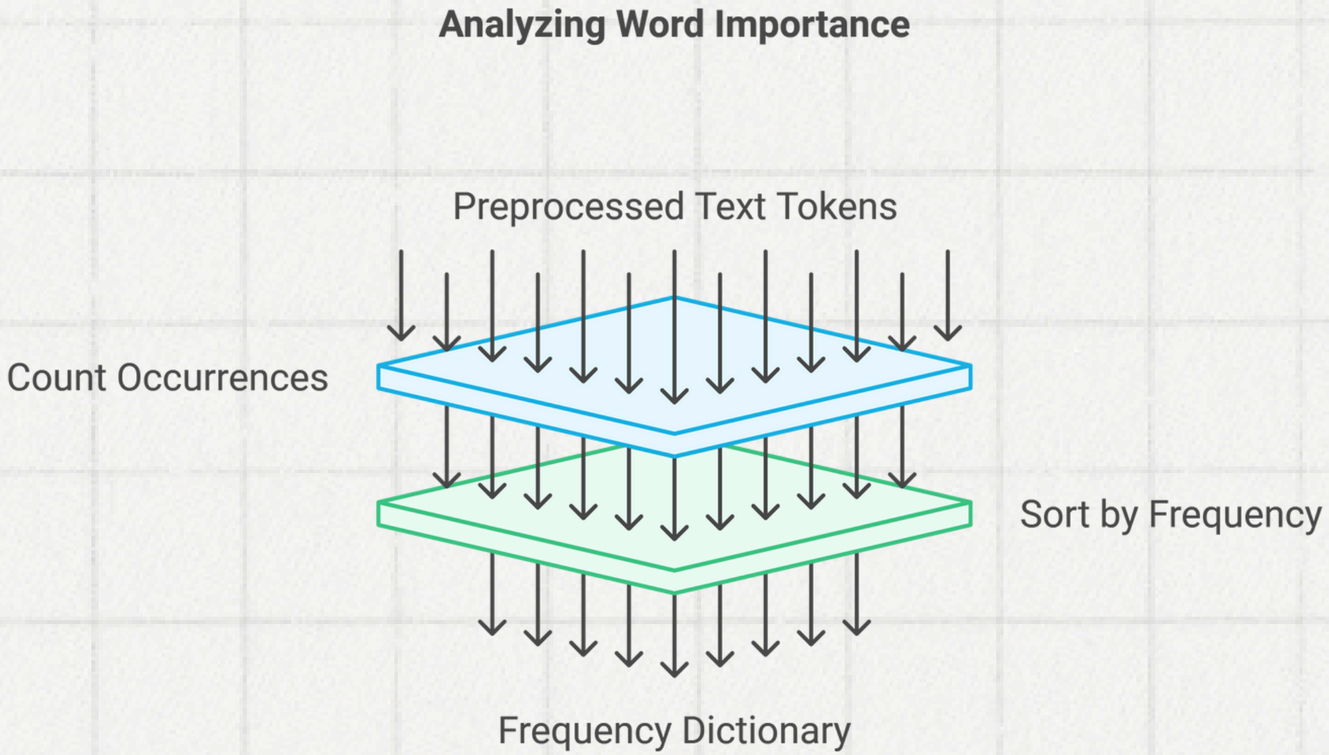
01- Text Preprocessing

To prepare the texts, we need to preprocess them. What we mean by preprocessing is removing unwanted characters, converting it to lowercase, and expanding contractions.

Decontract() + Preprocess() in the code

Vocabulary Building

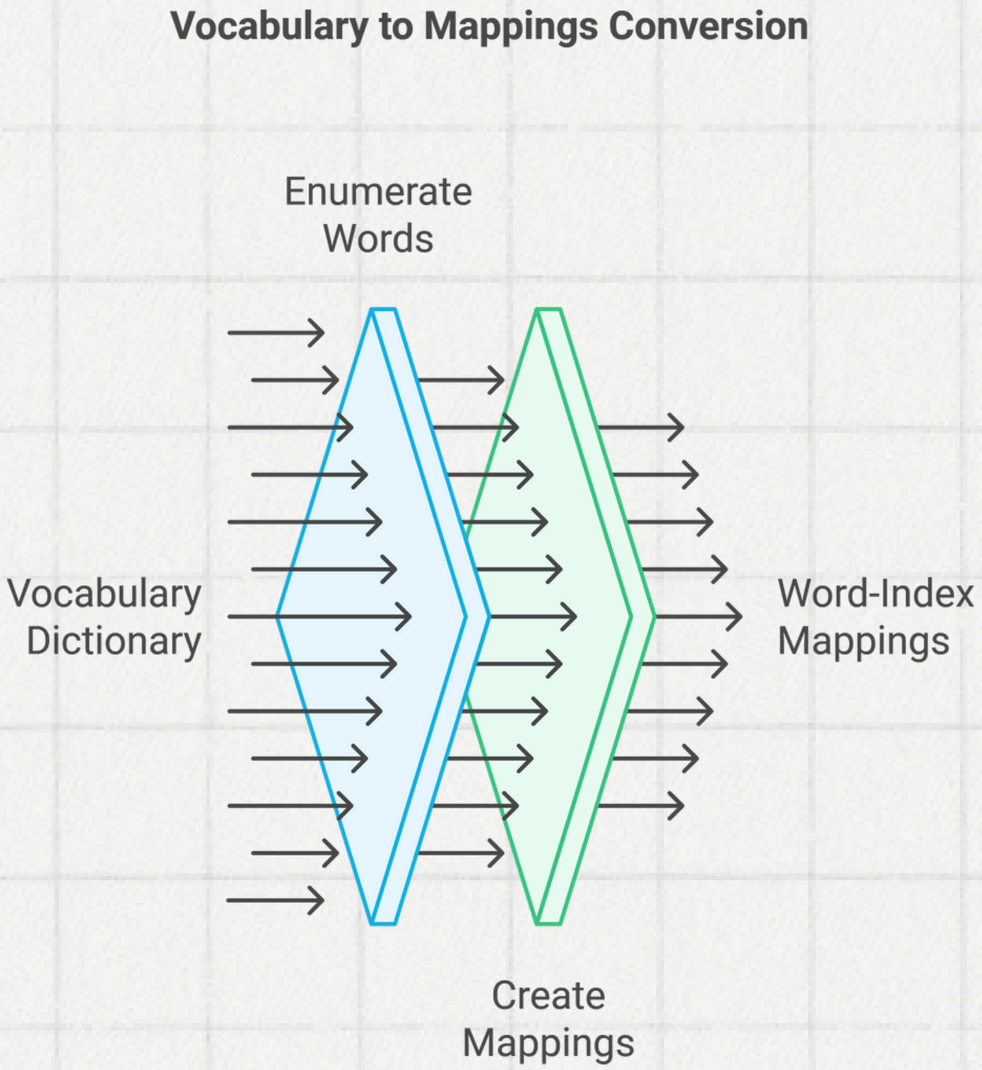
Analyze text to determine word importance based on frequency to provide a foundation for numerical representation in later steps.



word_frequency() in the code

Word Mapping

Transforms the vocabulary of words into numerical representations based in indexes.



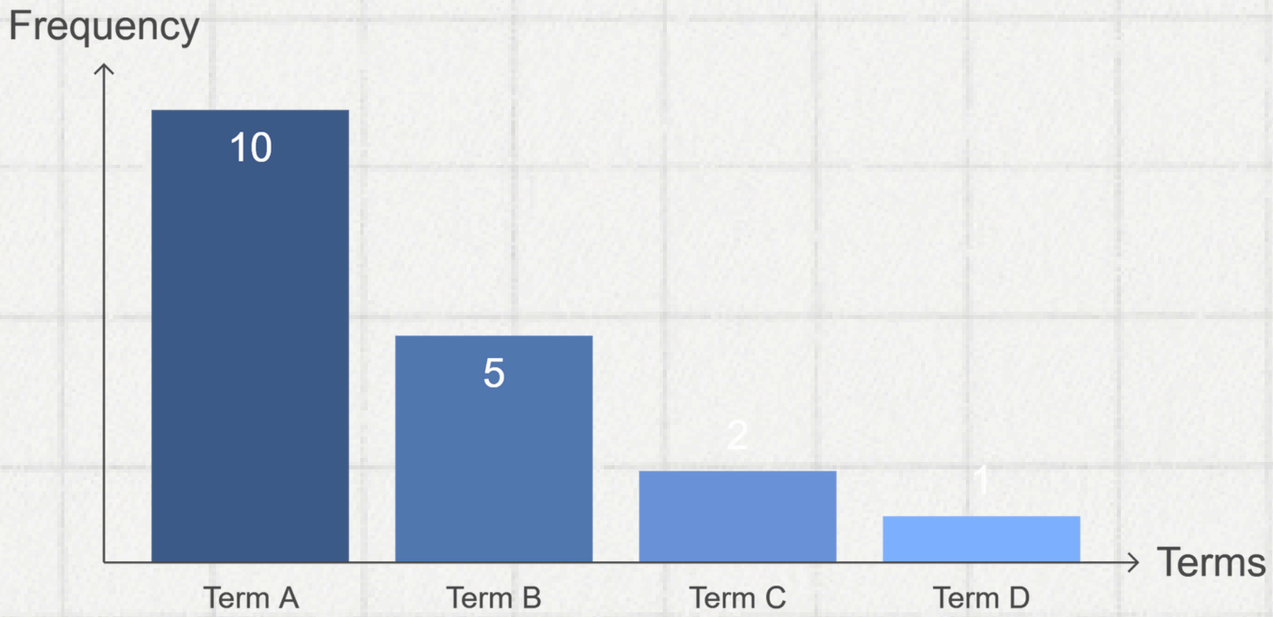
create_mapping() in the code

02- TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique used in Natural Language Processing (NLP) to convert text data into numerical representations while emphasizing important words in a document relative to the entire dataset.

a-Term Frequency [TF]

This measures how frequently a term appears in a document. The more times a term appears, the more important it is assumed to be.

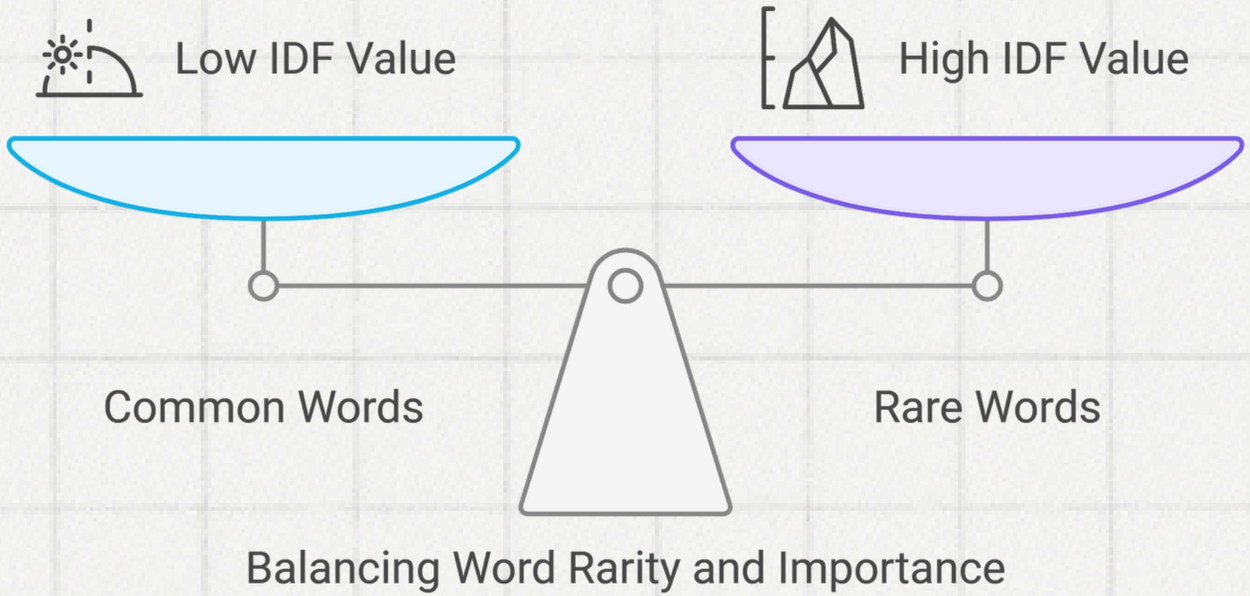


Term Frequency in Documents

computer_tf() in the code

b- Inverse Document Frequency [IDF]

This measures how important a term is across the entire corpus. So the rare words have high IDF while the frequently used words have low IDF. It helps to reduce the weight of common terms that appear in many documents.



compute_idf() in the code

c-TF-IDF

$$TFIDF\ score\ for\ term\ i\ in\ document\ j = TF(i,j) * IDF(i)$$

where

$IDF = Inverse\ Document\ Frequency$

$TF = Term\ Frequency$

$$TF(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}}$$

$$IDF(i) = \log_2 \left(\frac{\text{Total documents}}{\text{documents with term i}} \right)$$

and

$t = Term$

$j = Document$

- Words important in a document and rare across the corpus get high TF-IDF scores.
- Common words across documents are de-emphasized.

compute_tfidf() in the code

then we apply word mapping to go from dictionaries to vectors...

03- Cosine Similarity

Cosine Similarity is a measure of similarity between two vectors based on the cosine of the angle between them.

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

cosine_similarity() in the code

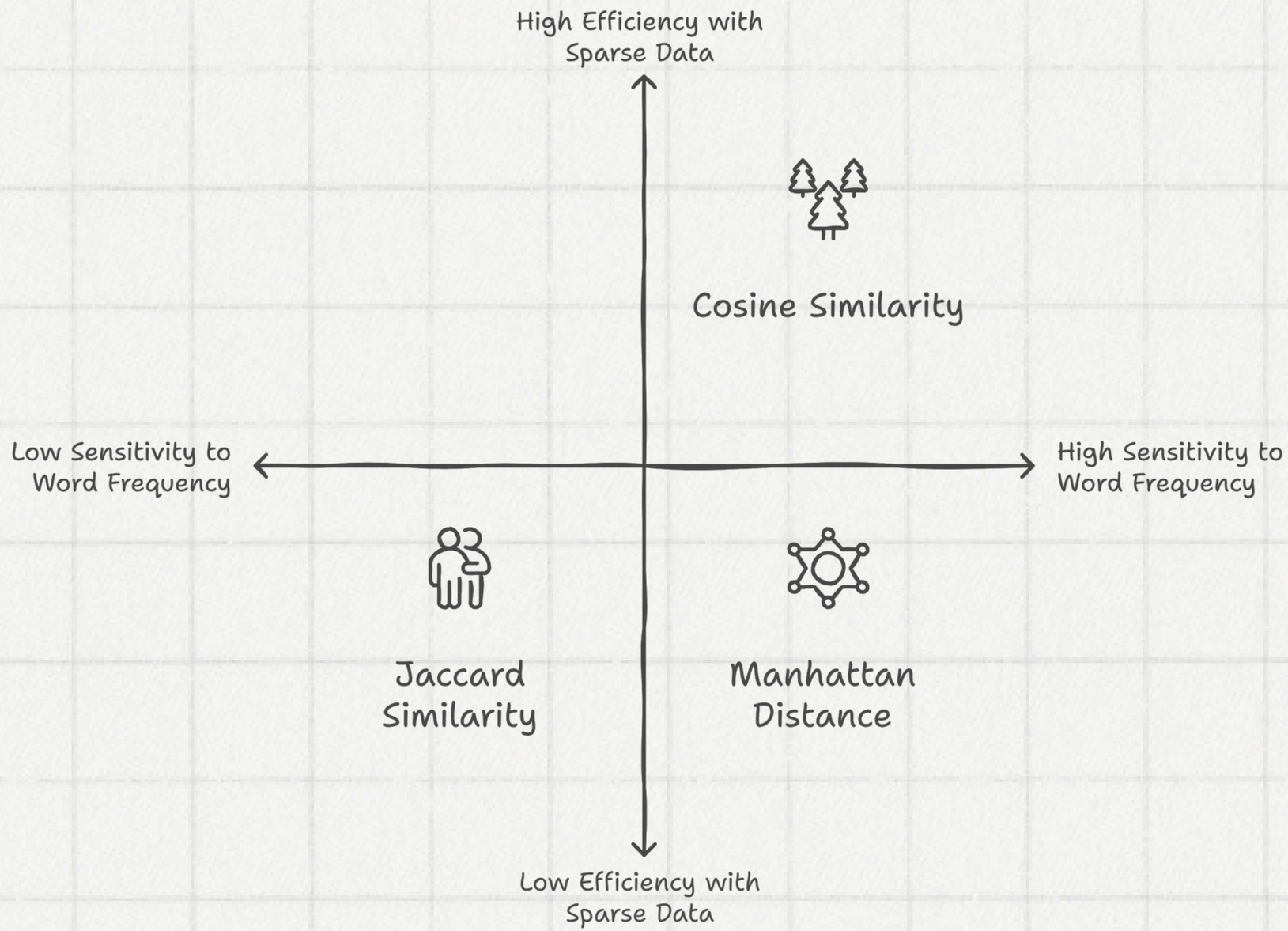
$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|}$$

- $A \cdot B$ is the dot product of vectors A and B.
- $\|A\|$ and $\|B\|$ are the magnitudes (Euclidean norms) of vectors A and B respectively.

Output: A value between -1 and 1:

- 1: Perfect similarity (texts are identical in meaning).
- 0: No similarity (texts are orthogonal).
- -1: Completely dissimilar (opposite meaning, rare in text applications).

Comparative Analysis of Text Similarity and Distance Metrics



Thank you very much!

check the code!

https://github.com/nweera/TextContractionExpander