Advanced Text Representation

Lecture 3: TF-IDF

Harito

September 15, 2025

Recap: The Problem with Count Vectors

Let's look at our corpus from last time:

- D1: "I love NLP."
- D2: "I love programming."
- D3: "NLP is a subfield of AI."

The term ‘"I"‘ appears in two documents, and so does ‘"NLP"‘. But intuitively, ‘"NLP"‘ is more descriptive of a document's topic than ‘"I"‘.

Problem: Raw counts treat all words equally. Very frequent words (like 'the', 'is', 'a') can dominate, while rare, more informative words are treated as less important.

Introducing TF-IDF

Solution

We need a measure that balances a term's frequency in a document with its frequency across the entire corpus. This measure is **TF-IDF**.

TF-IDF stands for **Term Frequency-Inverse Document Frequency**.

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

- **TF**: How important is the term in this *one document*?
- **IDF**: How important is the term in the *entire corpus*?

Term Frequency (TF)

This is the easy part - it's exactly what we calculated in 'CountVectorizer'!

Term Frequency is the raw count of a term 't' in a document 'd'.

$$\text{tf}(t, d) = \text{count of } t \text{ in } d$$

Example (D1: "I love NLP.")

- tf('i', D1) = 1
- tf('love', D1) = 1
- tf('nlp', D1) = 1

Inverse Document Frequency (IDF)

This is the magic ingredient. It measures how informative a word is.

Inverse Document Frequency is a logarithmic measure of how rare a term is across the corpus.

$$\text{idf}(t) = \log \frac{N}{df_t + 1}$$

- N: Total number of documents in the corpus.
- df_t: Document Frequency - the number of documents containing the term ‘t‘.
- $+1$: ”Smoothing” to avoid division by zero if a term is not in the corpus.

A high IDF means a term is rare. A low IDF means a term is common.

Example: Calculating IDF

Corpus (N=3):

- D1: "I love NLP."
- D2: "I love programming."
- D3: "NLP is a subfield of AI."

Document Frequencies (df):

- df('i') = 2
- df('love') = 2
- df('nlp') = 2
- df('programming') = 1
- df('is') = 1
- ...and so on for other words.

Example: Calculating IDF

Corpus (N=3):

- D1: "I love NLP."
- D2: "I love programming."
- D3: "NLP is a subfield of AI."

Document Frequencies (df):

- df('i') = 2
- df('love') = 2
- df('nlp') = 2
- df('programming') = 1
- df('is') = 1
- ...and so on for other words.

IDF Calculations:

- idf('i') = log(3 / (2+1)) = log(1) = 0
- idf('love') = log(3 / (2+1)) = 0
- idf('nlp') = log(3 / (2+1)) = 0
- idf('programming') = log(3 / (1+1)) = log(1.5) = 0.405

Notice how the common words ('i', 'love', 'nlp') get a low IDF score!

Putting It All Together: TF-IDF

$$\text{tfidf}(t, d) = \text{tf}(t, d) \times \text{idf}(t)$$

Let's calculate the scores for **D2: "I love programming."**

- tf('i', D2) = 1
- tf('love', D2) = 1
- tf('programming', D2) = 1

Putting It All Together: TF-IDF

$$\text{tfidf}(t, d) = \text{tf}(t, d) \times \text{idf}(t)$$

Let's calculate the scores for **D2: "I love programming."**

- tf('i', D2) = 1
- tf('love', D2) = 1
- tf('programming', D2) = 1
- tfidf('i', D2) = 1 * idf('i') = 1 * 0 = 0
- tfidf('love', D2) = 1 * idf('love') = 1 * 0 = 0
- tfidf('programming', D2) = 1 * idf('programming') = 1 * 0.405 = 0.405

The resulting TF-IDF vector for D2 would give weight only to the term 'programming'.

L2 Normalization

It is common practice to normalize the TF-IDF vectors so that they all have a length (Euclidean norm) of 1.

Why?

It prevents longer documents from having a higher similarity score just because they are long and have higher term counts. It ensures all vectors are on a comparable scale.

$$V_{norm} = \frac{V}{\|V\|_2}$$

Where $\|V\|_2$ is the Euclidean norm (square root of the sum of the squared elements).

Next Steps

Time for Lab 3!

Objective:

- Implement a ‘TfidfVectorizer‘.
- See how it produces more meaningful text representations than raw counts.