Introduction to Text Mining and Natural Language Processing

Session 4: Documents as Vectors

Hannes Mueller

January 2026

Barcelona School of Economics

Orientation

Course Overview

Part 1: Project Design and Getting the Text (Session 2)

Part 2: Text Mining Basics (Sessions 3 and 4)

Part 3: Dimensionality Reduction (Sessions 5 to 7)

In-class assignment in session 6

Part 4: Supervised Learning with Text (Sessions 8 and 9)

In-class assignment in session 9

Write me your term-paper ideas throughout.

Term paper presentations: 16th of March (feedback!)

Vectorizing

Vectorizing (Reminder)

After pre-processing, each document is a finite list of terms (unigrams, bigrams...). This gives us a vocabulary. In the example above this looks something like this:

$\{'palestinian' : 0,' president' : 1,' mahmoud' : 2,' abbas' : 3,' said' : 4,' wednesday' : 5,' late' : 6,' former' : 7,' israeli' : 8,' shimon' : 9,' peres' : 10,' exerted' : 11,' unremitting' : 12,' efforts' : 13,' make' : 14,' peace' : 15,' last' : 16,' moment' : 17,' life' : 18,' reach' : 19,' permanent' : 20,' since' : 21,' oslo' : 22,' agreement' : 23,' signed' : 24,' israel' : 25,' 1993' : 26,' condolence' : 27,' letter' : 28,' family' : 29,' carried' : 30,' state' : 31,' news' : 32,' agency' : 33, ...$

Building the DTM

We then use this vocabulary to get to the corpus representation as a document term matrix:

- Index each unique term in the corpus by some $v \in \{1, ..., V\}$, where V is the number of unique terms.
- For each document $d \in \{1, ..., D\}$ compute the count $x_{d,v}$ as the number of occurrences of term v in document d.

The $D \times V$ matrix X of all such counts is called the document-term matrix.

DTM View

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & \cdots & x_{1V} \\ x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & \cdots & x_{2V} \\ x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & \cdots & x_{3V} \\ x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & \cdots & x_{4V} \\ x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & \cdots & x_{5V} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{D1} & x_{D2} & x_{D3} & x_{D4} & x_{D5} & \cdots & x_{DV} \end{bmatrix}$$

This DTM is a $D \times V$ matrix.

Introduction to CountVectorizer

Command Syntax

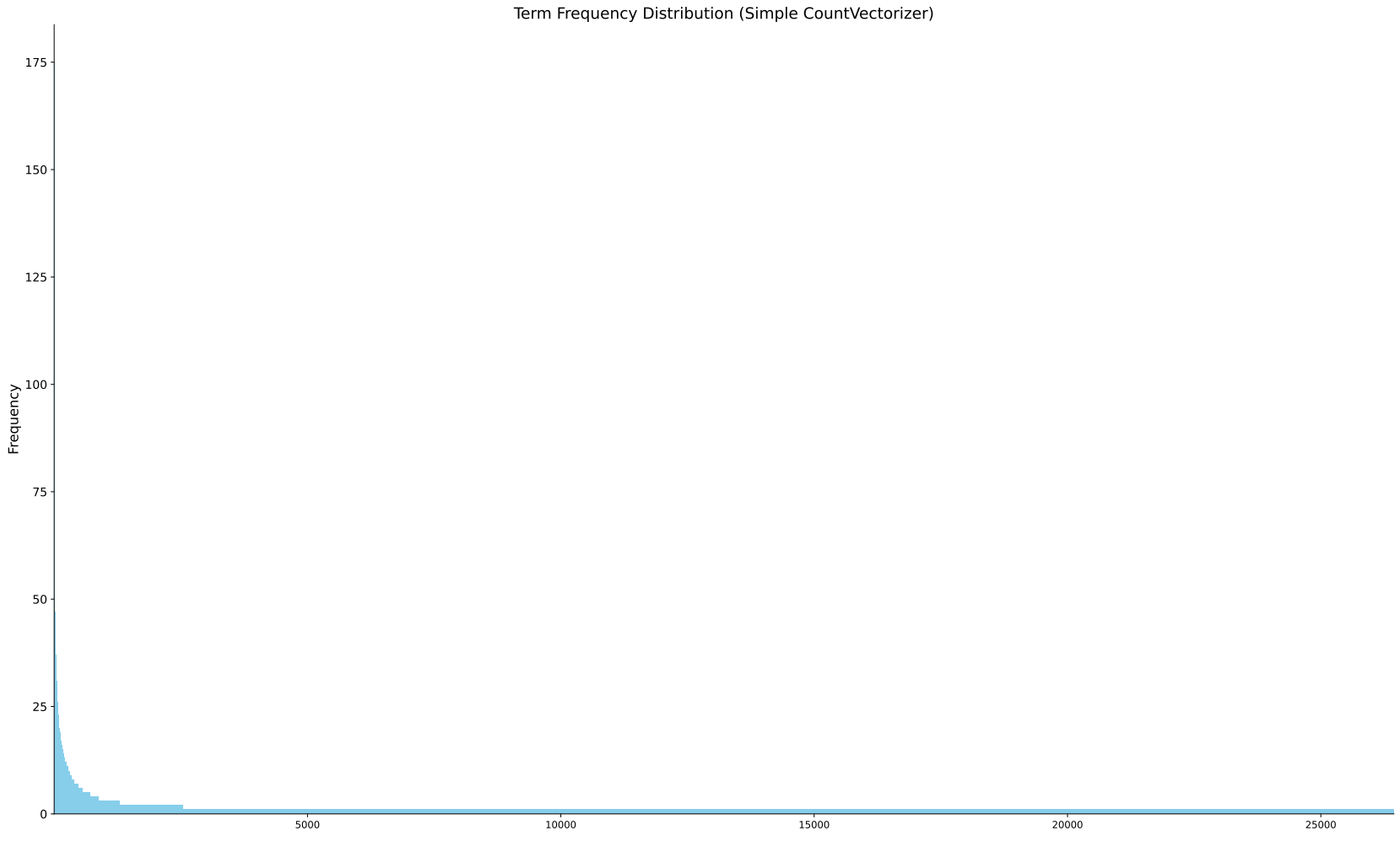
```
CountVectorizer(ngram_range=(1, 3),
                min_df=0.05, max_df=0.3)
```

Parameters:

- **ngram_range=(1, 3):** Considers unigrams, bigrams, and trigrams.
- **min_df=0.05:** Ignores terms appearing in fewer than 5% of documents.
- **max_df=0.3:** Ignores terms appearing in more than 30% of documents.

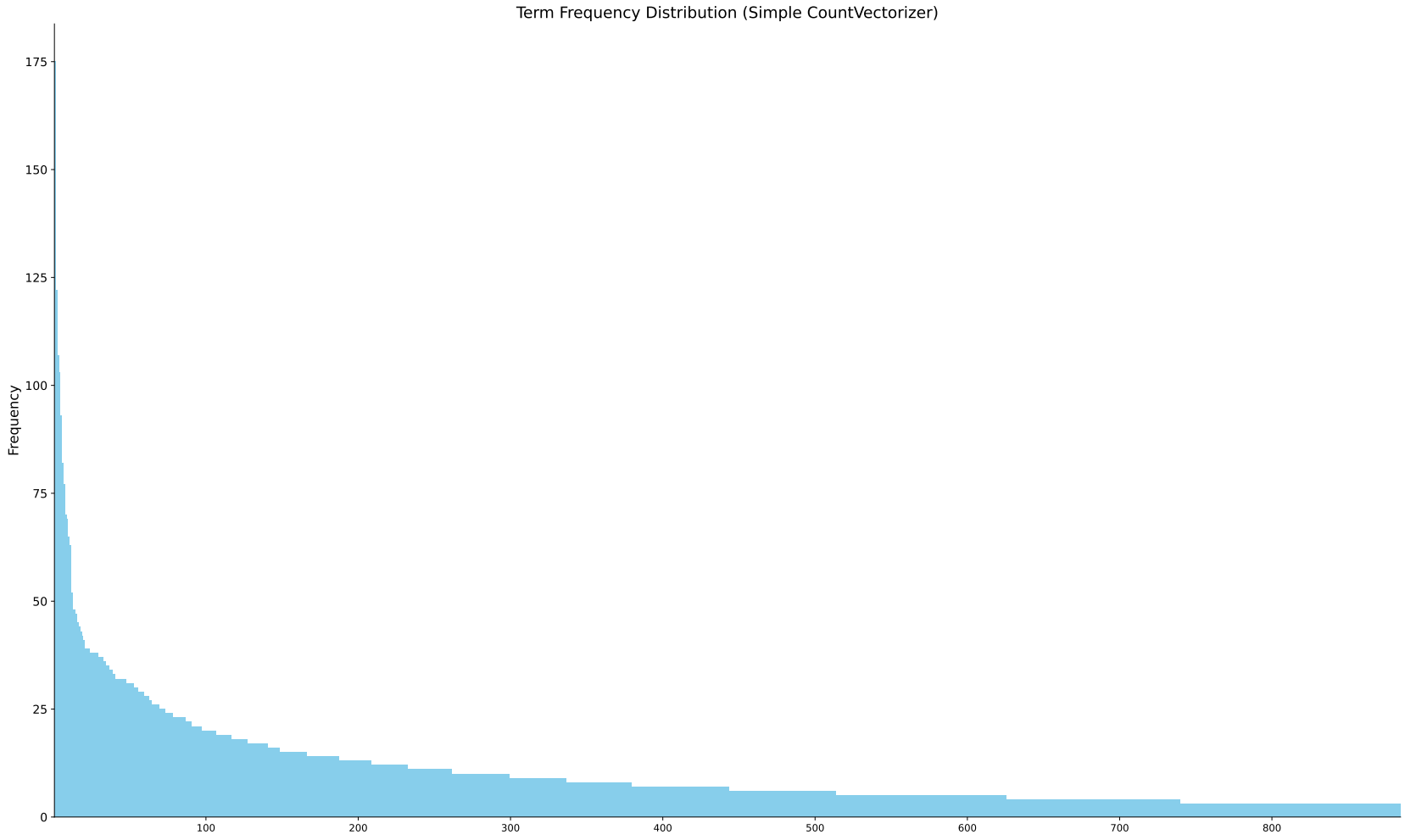
Think about what this does to the term distribution.

Term Frequency Distribution



Let's use min_df to cut off tail.

Term Frequency Distribution (min_df used)



Inputs and Outputs of CountVectorizer

Input

List of text documents (strings).

Outputs

Scipy sparse matrix: each row represents a document, and each column represents a term from the vocabulary. The values indicate the frequency of the term in the document.

Vocabulary: maps each term to its corresponding feature index in the document-term matrix.

Method: `get_feature_names_out()` retrieves the list of terms in the vocabulary, ordered by their feature indices.

Text as Vectors: the Fun Begins

Some thoughts about term frequencies

- We can get some representation of a document through term frequencies, $x_{d,v}$. Three common problems with this:
- 1) some documents are much longer than other documents. What should we do?
- Normalize counts by document length.
  - Let $x_{d,v}$ be the count of the term v in document d.
  - Calculate $f_{d,v} = x_{d,v}/(\sum_v x_{d,v})$
  - Keep this in mind also when aggregating documents!
- 2) some terms that are very common across the corpus.
- 3) sometimes we want to weigh specific content stronger

Tf-idf

Problem 2) Zipf's Law

- Zipf's Law is an empirical regularity for most natural languages:
  - Given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc.
- Means that a few terms will have very large counts, many terms have small counts.

How to Re-Weigh Counts

- Imagine you want to get to a vector representation of a document that controls for this.
- Let $x_{d,v}$ be the count of the term v in document d.
- We can express this count in the standard way as an absolute count $x_{d,v}$ or as

$$tf_{d,v} = \begin{cases} 0 & \text{if } x_{d,v} = 0 \\ \log(x_{d,v}) + 1 & \text{otherwise} \end{cases}$$

  which is (confusingly) called the term frequency.

Inverse Document Frequency

- Standard method in weighting text.
- Idea: give less weight to terms that appear in a lot of documents.
- Sometimes this is even used in pre-processing text.
- $idf_v = \log(D/df_v)$ where
  - df_v is the number of documents that contain the term v
  - D is the number of documents
- Higher weight to words in fewer documents.
- Log dampens effect of weighting.
- A *smooth* alternative is $idf_v = \log\left(\frac{D}{df_v}\right) + 1$

Now Build the Score

- tf-idf score is situated at the document (d) - term (v) level.

$$tf\text{-}idf_{d,v} = \begin{cases} 0 & if\ x_{d,v} = 0 \\ [1 + \log(x_{d,v})] * \log(D/df_v) & otherwise \end{cases}$$

- sklearn Tf-idf vectorizer does instead $tf\text{-}idf_{d,v} = x_{d,v} * (\log[(1+D)/(1+df_v)] + 1)$ but it also has the option of imposing sub-linearity

Why prefer sublinear TF?

If you expect that **large counts should be heavily rewarded**, then using raw TF might be appropriate. For example, if your domain suggests that the more times a term appears, the more relevant that document is (with no upper limit), raw TF might be beneficial.

If you worry that **repeated mentions beyond a point stop adding real topical relevance**, sublinear TF helps reduce the overemphasis. In many text domains, once a term appears more than a handful of times, it's already signaling the document is about that term. Additional mentions might be near-duplicates (like repeated headings) or boilerplate text. Sublinear TF tries to keep that from overshadowing other terms.

Example: A Small Document-Term Matrix

Assume $D = 5$ documents and a vocabulary of $V = 7$ terms:

$$\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$

The document-term matrix of raw counts is:

$$X = \begin{bmatrix} 2 & 0 & 1 & 0 & 3 & 0 & 0 \\ 1 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 2 & 2 & 2 \\ 4 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Key design features:

- Column v_1 appears in 4 out of 5 documents ($df_1 = 4$) $\Rightarrow$ should be **downweighted**.
- Column v_7 appears only in document $d = 3$ with count 2 ($df_7 = 1$) $\Rightarrow$ should be **upweighted**.

Document Frequencies and IDF (see the contrast)

Define:

$$df_v = \sum_{d=1}^{D} \mathbb{I}(x_{d,v} > 0)$$

From the matrix X:

$$(df_1, df_2, df_3, df_4, df_5, df_6, df_7) = (4, 2, 2, 2, 4, 2, 1)$$

Standard IDF: $idf_v = \log\left(\frac{D}{df_v}\right)$

In particular:

$$idf_1 = \log\left(\frac{5}{4}\right) \approx 0.223, \qquad idf_7 = \log\left(\frac{5}{1}\right) = \log(5) \approx 1.609$$

So the rare term v_7 gets an IDF about $\frac{1.609}{0.223} \approx 7.2$ times larger than the common term v_1.

TF–IDF Matrix (standard: sublinear TF, standard IDF)

The matrix

$$X = \begin{bmatrix} 2 & 0 & 1 & 0 & 3 & 0 & 0 \\ 1 & 1 & 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 2 & 2 & 2 \\ 4 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

with $tf\text{-}idf_{d,v} = [1 + \ln(x_{d,v})] * \ln(D/df_v)$ becomes

$$X = \begin{bmatrix} 0.378 & 0 & 0.916 & 0 & 0.468 & 0 & 0 \\ 0.223 & 0.916 & 0 & 0 & 0.378 & 0 & 0 \\ 0.223 & 0 & 0 & 0.916 & 0.378 & 1.551 & 2.725 \\ 0.532 & 0 & 0.916 & 0 & 0 & 0.916 & 0 \\ 0 & 1.551 & 0 & 0.916 & 0.223 & 0 & 0 \end{bmatrix}$$

A Detour: Similarity

Similarity

- Call the vector representation of a document, d, $\vec{V}_d$, where the entries of the vector can be the term frequencies $x_{d,v}$, something based on a dictionary, or the $tfidf_{d,v}$.
- Imagine you want to measure whether two documents are similar using this vector.
- One way is to calculate the euclidian distance between two documents 1 and 2 is:

$$d_{1,2} = \sqrt{\sum_v \left(x_{d1,v} - x_{d2,v}\right)^2}$$

- Why can this backfire badly?

Cosine similarity

Instead, what we typically do is to calculate the cosine similarity.

$$cs_{1,2} = \frac{\vec{V}_1 \cdot \vec{V}_2}{\left|\vec{V}_1\right|\left|\vec{V}_2\right|}$$

Think of this as a two-step procedure. First, if we normalize vectors by their length

$$\vec{v}_d = \frac{\vec{V}_d}{\left|\vec{V}_d\right|}$$

so they have length 1 and then we look at the dot product.

$$cs_{1,2} = \vec{v}_1 \cdot \vec{v}_2$$

Why the dot product?

- You might ask yourself - is the dot product a reasonable measure of similarity/distance?
- It tends to give a lot of weight to large values.
- Important coinciding counts will give a lot of weight to large entries in the vectors $\vec{V}$.
- Smaller entries will receive less attention.
- But there is an important fact "behind the scenes" that makes the dot product surprisingly reasonable as a similarity measure.

The dot product and Euclidean distance

Apart of the other anwers, there is indeed a relation between Euclidean distance $d(X, Y)$ and the inner product $\langle X, Y \rangle$ if the vectors $X, Y \in \mathbb{R}^N$ are normalized, that is $\langle X, X \rangle = \langle Y, Y \rangle = 1$, in particular

$$\frac{d(X, Y)^2}{2} = 1 - \langle X, Y \rangle.$$

This can be shown as

$$d(X, Y)^2 = \langle X - Y, X - Y \rangle = \langle X, X \rangle + \langle Y, Y \rangle - 2\langle X, Y \rangle = 2(1 - \langle X, Y \rangle)$$

Share Cite Follow

edited Dec 18, 2019 at 8:10

answered Nov 2, 2018 at 15:48

Pavel Prochazka

341 ■ 2 ▲ 7

Centroid

The centroid of a collection or cluster of documents ω is given by

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{i \in \omega} \vec{v}_i$$

Recap

- We now have a good idea of vector representations of documents.
- We have seen two types:
  - Absolute and relative term frequency counts
  - Tf-idf counts
- Important: it helps a lot to think about this carefully in your application. Try your own version!
- After the coding session we will see dictionaries as the third (very radical) type of weighting.
- Next session we will explore methods that exploit these vector representations.
- We will also explore a "weird" weighting to practice.

Application: Mark Carney's WEF speech

https://paulwells.substack.com/p/the-carney-doctrine

Dictionary Methods

Dictionary Methods

- Remember that the basic problem we are trying to solve is that the number of terms, V, is relatively high.
- Challenge in using text data for decision making is to therefore to reduce its dimensionality down from V.
- Dictionary methods are typically used in two cases:
  - 1) Human has a strong prior on what to use, i.e. specific keywords.
  - 2) Human knows what "kind" of text they want to capture, i.e. text which talks about finance.
- 1) is trivial. Key difficulty is how to come up with a dictionary for 2). We will do mostly literature discussion today.

Dictionary Method: Overview

- What is it?
- How has research derived dictionaries and used the resulting counts to capture specific concepts?
- Implementation of one example - partially left for small homework.

Dictionary Based Method

- Dictionary is a list of terms. Call this set of terms $\mathfrak{D}$.
- Boolean search provides a count of the number of times specific terms appear in a document, $x_{d,v}$.
- Important advantage: often you can query a database that you don't own to give you $x_{d,v}$
- In most methods, this is then aggregated to deliver some sort of score or index at document level (which is then typically further aggregated).
- In applications both the dictionaries vary widely and the ways to score documents vary.

Aggregation Methods

- Aggregation is as important as the dictionaries themselves.
- Simple sum: score d with $x_d = \sum_{v \in \mathfrak{D}} x_{d,v}$
- Why do we typically not use (aggregates of) these raw counts?
- Normalized sum: score d with $s_d = \sum_{v \in \mathfrak{D}} x_{d,v} / \sum_v x_{d,v}$.
- Indicator: score d with $I\left(\sum_{v \in \mathfrak{D}} x_{d,v} > 0\right)$
- Interaction: score d with $\prod_{v \in \mathfrak{D}} I\left(x_{d,v} > 0\right)$