ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI FORLÌ

91258 / B0385
Natural Language Processing

Lesson 6. Term Frequency–Inverse Document Frequency

Alberto Barrón-Cedeño
a.barron@unibo.it

15/10/2025

Table of Contents

Previously

- Pre-processing
- BoW representation
- One rule-based sentiment model
- One statistical model (Naïve Bayes)

Table of Contents

These slides cover roughly chapter 3 of Lane et al. (2019)

From BoW to term frequency

Intuition

1. The frequency of a token t in a document d is an important factor of its relevance
2. The relative frequency of a word in a document wrt all other documents in the collection provides even better information

Binary Bag of Words

We departed from a binary representation

We were simply interested in the existence (or not) of a word in a document:

$$\begin{array}{cc} & \begin{array}{ccccccccccccc} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & w_9 & w_{10} & w_{11} & w_{12} & w_{13} \end{array} \\ d_1 = & \left[\begin{array}{ccccccccccccc} 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \\ d_2 = & \left[\begin{array}{ccccccccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array} \right] \end{array}$$

"Counting" Bag of Words

A word that appears often contributes more to the "meaning" of the document

A document with many occurrences of "good", "awesome", "best" is more positive than one in which they occur only once

$$\begin{array}{ccc} d_1 & = & [\ 0\ 1\ 0\ 0\ 2\ 0\ 1\ 3\ 0\ 0\ 0\ 0\ 0\ ] \\ d_2 & = & [\ 2\ 3\ 5\ 0\ 0\ 0\ 0\ 4\ 0\ 0\ 0\ 4\ 2\ ] \end{array}$$

Let us see...

Already a useful representation for diverse tasks, such as detecting spam and computing "sentiment"

tf: Term Frequency

tf represents the number of times a word appears in a document

(In general) the frequency of a word depends on the length of the document

- Shorter document → lower frequencies
- Longer document → higher frequencies

Ideally, our *counting* should be document-length independent.

Normalisation!

tf: Term Frequency (Normalised)

Why normalising?

Example

word dog appears 3 times in d_1

word dog appears 100 times in d_2

Intuition: dog is way more important for (representative of) d_2 than for (of) d_1

d_1 is an email by a veterinarian (300 words)

d_2 is *War & Peace* (580k words)

If normalised...

$$
\begin{aligned}
tf(\text{dog}, d_1) &= 3/300 &= 0.01 \\
tf(\text{dog}, d_2) &= 100/580,000 &= 0.00017
\end{aligned}
$$

Reminder: normalised frequencies can be considered probabilities

tf: Term Frequency (Normalised)

Playing with a longer text

- Loading frequencies into a dictionary
- Vectorising frequencies
- Normalising frequencies

tf: Term Frequency

From a single to multiple documents

- The vectors have to be comparable across documents → normalisation
- Each position in the vectors must represent the same word

This is when representations become sparse: a matrix packed with 0

Sparse vector: most of the elements are zero

Dense vector: most of the elements are non-zero

Let us see

See `https://en.wikipedia.org/wiki/Sparse_matrix`

Vectors of Term Frequency

Vectors

- Primary building blocks of linear algebra
- Ordered list of numbers, or coordinates, in a vector space
- They describe a location in that space...
  or identify a direction/magnitude/distance in that space

Vector space Collection of all possible vectors

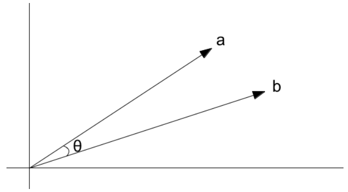
$[1, 4] \rightarrow$ 2D vector space
$[1, 4, 9] \rightarrow$ 3D vector space

We have an 18D vector space (we have seen $20k$+D ones!)

Comparing Vectors

Cosine similarity

The cosine of the angle between two vectors (θ theta)



$$\cos\theta = \frac{a \cdot b}{|\,a\,||\,b\,|} \quad (1)$$

where

$a \cdot b$ is the dot product (we know it)
$|\,a\,|$ is the magnitude of vector a

Let us see an implementation (but there are efficient libraries to do it)

Comparing Vectors

Cosine similarity

Properties of the cosine similarity

- It is ranged in $[-1, 1]$
  this is a very convenient range for ML
- $cos = 1$: identical normalised vectors that point in exactly the same direction
- $cos = 0$: two orthogonal vectors (no components shared)
- $cos = -1$: two opposite vectors
- In *tf*-like representations, cosine is ranged in $[0, 1]$
  (no negative frequencies)

Zipf's Law

Zipf's Law

Given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table.[1]

pos(w)	*freq*(w)
1st	k
2nd	$k/2$
3rd	$k/3$
...	...

The system behaves "roughly" exponentially

Examples of exponential systems: population dynamics and COVID-19

Let's see it for text

[1] George K. Zipf; 1930s

Zipf's Law

Frequencies of the Brown corpus: *expected* vs *actual*

w	$f_{exp}(w)$	$f_{act}(w)$
the	–	69,971
of	34,985	36,412
and	23,323	28,853
to	17,492	26,158
a	13,994	23,195
in	11,661	21,337
that	9,995	10,594
is	8,746	10,109
was	7,774	9,815
he	6,997	9,548
for	6,361	9,489
it	5,830	8,760
with	5,382	7,289
as	4,997	7,253
his	4,664	6,996

Zipf's Law

Stats

- This distribution only holds with large volumes of data (not in a sentence, not in a couple of texts)
- By computing this distribution, we can obtain an *a priori* likelihood that a word w will appear in a document of the corpus

Inverse Document Frequency

idf–Inverse Document Frequency

Two ways (among many others) to count tokens

tf per document

idf across a full corpus

Let's see. . .

IDF How strange is it that this token appears in this document?

If w appears in d a lot, but rarely in any other $d' \in D \mid d' \neq d$
w is quite important for d

Let's see

IDF and Zipf

Let us assume a corpus D, such that $|D| = 1M$

- 1 document $d \in D$ contains "cat"
  $idf(cat) = 1,000,000/1 = 1,000,000$
- 10 documents $\{d_1, d_2, \ldots, d_{10}\} \in D$ contain "dog"
  $idf(dog) = 1,000,000/10 = 100,000$

According to Zipf's Law, when comparing w_1 and w_2, even if $f(w_1) \sim f(w_2)$, one will be exponentially higher than the other one!

We need the inverse of $exp()$ to mild the effect: $log()$

$$
\begin{aligned}
idf(cat) &= log(1,000,000/1) &= log(1,000,000) &= 6 \\
idf(dog) &= log(1,000,000/10) &= log(100,000) &= 5
\end{aligned}
$$

tf-idf

$$tf(t,d) = \frac{count(t,d)}{\sum_t count(t,d)} \quad (2)$$

$$idf(t,D) = log \frac{\text{number of documents in } D}{\text{number of documents in } D \text{ containing } t} \quad (3)$$

$$tfidf(t,d,D) = tf(t,d) * idf(t,D) \quad (4)$$

- The more often t appears in d, the higher the TF (and hence the TF-IDF)
- The higher the number of documents containing t, the lower the IDF (and hence the TF-IDF)

tf-idf

Outcome The importance of a token in a specific document given its usage across the entire corpus.

"TF-IDF, is the humble foundation of a simple search engine"
(Lane et al., 2019, p. 90)

Let's see

tf-idf Implementation

- We "hand-coded" the *tf-idf* implementation
- Optimised and easy-to-use libraries exist
- `scikit-learn` is a good alternative[2]

Let us see

[2] `http://scikit-learn.org`. As usual, install it the first time; e.g., `pip install scipy; pip install sklearn`

tf-idf

Final Remarks

tf-idf-like weighting...

- is the most common baseline representation in NLP/IR papers nowadays
- is in the core of search engines and related technology
- Okapi BM25 has been one of the most successful ones (Robertson and Zaragoza, 2009)
  - Okapi First system using BM25 (U. of London)
  - BM best matching
  - 25 Combination of BM11 and BM15
- Cosine similarity is a top choice metric for many text vector representations.
- Nothing prevents you from weighting n-grams, for $n = [1, 2, \ldots]$

Coming Next

- Towards "semantics"

References

Lane, H., C. Howard, and H. Hapkem
2019. *Natural Language Processing in Action*. Shelter Island, NY: Manning Publication Co.

Robertson, S. and H. Zaragoza
2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333—-389.