TEXT REPRESENTATION

Gabriella Pasi
pasi@disco.unimib.it

Simplest way : Binary term-document weighting. Example by incidence matrix

Documents

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

↑ ***Vocabulary V***

Each document can be ***represented*** by a set of terms **or** by a binary vector $\in \{0,1\}^{|V|}$

Term-document weighting
Count matrix

- Consider the number of occurrences of a term in a document:
  - Each document is a count vector in $\mathbb{N}^v$: a column below

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Bag of words model

- Vector representation does not consider the ordering of words in a document
- *John is quicker than Mary* and *Mary is quicker than John* have the same vectors
- This is called the bag of words model.
- We will see how to "recover" positional information

Bag-of-Words with N-grams

- N-grams: a contiguous sequence of N tokens from a given piece of text
  - E.g., '*Text mining is to identify useful information.*'
  - Bigrams: '*text_mining*', '*mining_is*', '*is_to*', '*to_identify*', '*identify_useful*', '*useful_information*', '*information_.*'
- Pros: capture local dependency and order
- Cons: a purely statistical view, increase the vocabulary size

Statistical properties of texts

- How is the frequency of different words distributed in a corpus?
- In natural language, there are a few very frequent terms and very few very rare terms.
- Zipf's law describes the frequency of an event (in our case a word) in a set according to its *rank* (rank: the numerical position of a word in a list sorted by decreasing frequency); Given a collection, sort the words w in decreasing order of their frequency $f(w)$ *in the collection* (with an increasing order of rank).

George Kingsley Zipf, *Human Behavior and the principle of least effort*, Addison Wesley, 1949.

Natural language and Zipf's law (1949)

Zipf's law: *the product of the frequency of use of words and the rank order is approximately constant.* So, the *frequency of w, f(w) is proportional to 1/r(w):*

$$f(w) \propto \frac{1}{r(w)} = \frac{K}{r(w)}$$

where *K* is a constant value. Different collections have different values of *K.*

Zipf G.K. *Human Behavior and the principle of least effort*, Addison Wesley, 1949.

Natural language and Zipf's law (1949)

- Given a collection, sort the words *w* in decreasing order of their frequency *f(w) in the collection* (with an increasing order of rank) :

Frequent Word	Number of Occurrences	Percentage of Total
the	7,398,934	5.9
of	3,893,790	3.1
to	3,364,653	2.7
and	3,320,687	2.6
in	2,311,785	1.8
is	1,559,147	1.2
for	1,313,561	1.0
The	1,144,860	0.9
that	1,066,503	0.8
said	1,027,713	0.8

Frequencies from 336,310 documents in the 1GB TREC Volume 3 Corpus
125,720,891 total word occurrences; 508,209 unique words

Zipf's law tells us

- Head words take large portion of occurrences, but they are semantically meaningless
  - E.g., the, a, an, we, do, to
- Tail words take major portion of vocabulary, but they rarely occur in documents
  - E.g., dextrosinistral

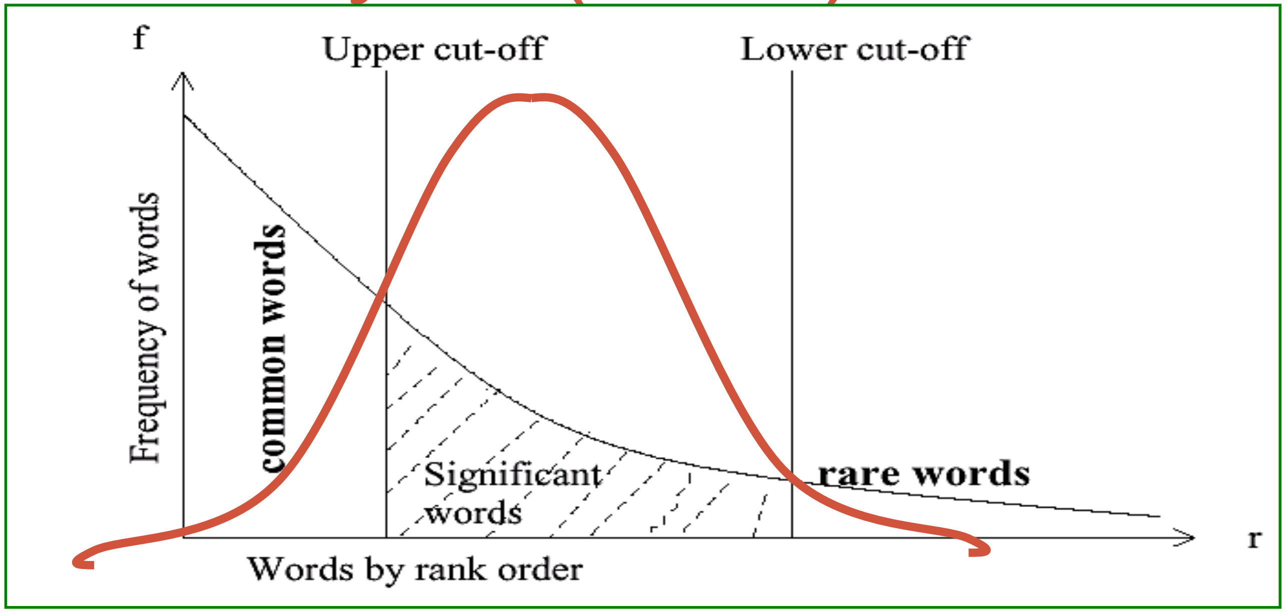
Luhn's Analysis (1958)

Not all words in a text describe the content with the same accuracy/informativity.

In 1958, Luhn noted that "*the frequency with which some words appear in a text provides an important indication of the significance of words. Moreover, the position of these words in sentences is another important parameter that indicates the significance of sentences*"

IDEA: association of weights to the terms that represent a document

Luhn's Analysis (1958)



- Discriminating power of significant words (Zipf's curve): the ability of words to discriminate the content of documents is maximum in the intermediate position between the two cut-off levels

Automatic document representation

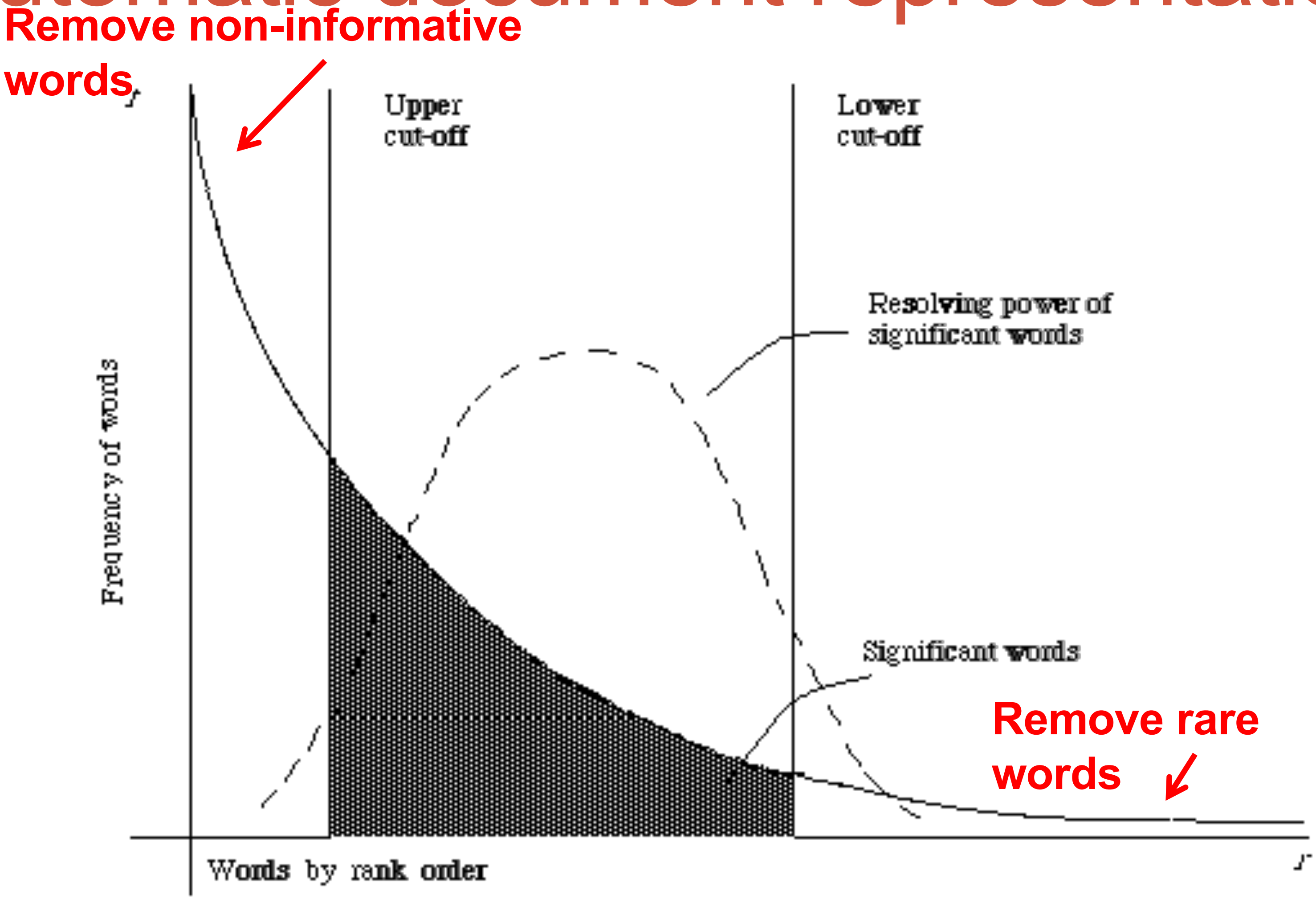


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adaped from Schultz[44] page 120)

Indexing criteria based on Luhn's Analysis

- Weighing the index terms: very frequent words assume a lower weight of significance
- Stop list: very frequent words are eliminated from the indexes (upper cut-off)
- Meaningful words: very frequent and infrequent words are eliminated from the indexes (upper and lower cut-off)

So: how to assign weights to terms ?

- Based on Luhn's analysis, proposals of term weighting appeared
- Two factors were identified:
  - Corpus-wise: some terms carry more information about the document content
  - Document-wise: not all terms are equally important
- How to measure them ?
  - Two basic heuristics
    - TF (Term Frequency) = Within-doc-frequency
    - IDF (Inverse Document Frequency)

Term frequency tf

- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- However, pure term frequency is not what we want:
  - A document with 10 occurrences of the term is more relevant than a document with 1 occurrence of the term.
  - But not 10 times more relevant.

A simple idea: term frequency adjusted for document length (the number of words in the document)

$$w_{t,d} = \frac{tf_{t,d}}{|d|}$$

Example

	d_1	d_2	d_3	d_4	d_5	d_6
	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

- $tf_{Antony,d_1} = ?$ 157
- $w_{Antony,d_1} = \frac{tf_{Antony,d_1}}{|d_1|} = ?$ $\frac{157}{157+4+232+57+2+2} = \frac{157}{454} = 0.34$

Normalizing by max occ

- To prevent a bias towards longer documents:

$$w_{t,d} = \frac{tf_{t,d}}{\max_{t_i \in d} tf_{t_i,d}}$$

- Where $\max_{t_i \in d} tf_{t_i,d}$ is the frequency of the most occurring term t_i in the document d.

Example

	d_1	d_2	d_3	d_4	d_5	d_6
	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

- $w_{t,d} = \frac{tf_{t,d}}{\max_{t_i \in d} tf_{t_i,d}} \rightarrow w_{Antony,d_1} = \frac{tf_{Antony,d_1}}{\max_{t_i \in d_1} tf_{t_i,d_1}} = ?$

Example

	d_1	d_2	d_3	d_4	d_5	d_6
	Antony and Cleopatra	**Julius Caesar**	**The Tempest**	**Hamlet**	**Othello**	**Macbeth**
Antony	**157**	**73**	**0**	**0**	**0**	**0**
Brutus	**4**	**157**	**0**	**1**	**0**	**0**
Caesar	**232**	**227**	**0**	**2**	**1**	**1**
Calpurnia	**0**	**10**	**0**	**0**	**0**	**0**
Cleopatra	**57**	**0**	**0**	**0**	**0**	**0**
mercy	**2**	**0**	**3**	**5**	**5**	**1**
worser	**2**	**0**	**1**	**1**	**1**	**0**

- $tf_{Antony,d_1} = ?$ 157
- $\max_{t_i \in d_1} tf_{t_i,d_1} = $ $tf_{Caesar,d_1} = ?$ 232

idf weight

- df_t is the document frequency of t: the number of documents that contain t.
  - df_t is an inverse measure of the informativeness of t
  - $df_t \leq N = |D|$

- We define the idf (inverse document frequency) of t by

$$idf_t = \log\left(\frac{N}{df_t}\right)$$

- We use $\log\left(\frac{N}{df_t}\right)$ instead of $\frac{N}{df_t}$ to «dampen» the effect of idf.

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (\mathrm{tf}_{t,d} / \max_{ti} \mathrm{tf}_{ti,d}) \times \log_{10}(N / \mathrm{df}_t)$$

- Best known weighting scheme in information retrieval
  - Note: the “-” in tf-idf is a hyphen, not a minus sign!
  - Alternative names: tf.idf, tf x idf
- Increases with the number of occurrences within a document
- Increases with the rarity of the term in the collection

Example – *df*

	d_1	d_2	d_3	d_4	d_5	d_6
	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

- $df_{Cleopatra} = ?$ 1
- $df_{worser} = ?$ 4

Example – *idf*

	d_1	d_2	d_3	d_4	d_5	d_6
	Antony and Cleopatra	**Julius Caesar**	**The Tempest**	**Hamlet**	**Othello**	**Macbeth**
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

- $idf_{Cleopatra} = \log\left(\frac{N}{df_{Cleopatra}}\right) = ?$ $\frac{6}{1} = 6$ $\log(6) = 0.78$
- $idf_{worser} = \log\left(\frac{N}{df_{worser}}\right) = ?$ $\frac{6}{4} = 1.5$ $\log(1.5) = 0.18$

tf-idf weighting

- The $tf\text{-}idf$ weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = \frac{tf_{t,d}}{\max_{t_i \in d} tf_{t_i,d}} \cdot \log\left(\frac{N}{df_t}\right)$$

- Note: the "-" in tf-idf is a hyphen, not a minus sign!

- Alternative names: $tf.idf$, $tf \times idf$

Example *tf-idf*

	d_1	d_2	d_3	d_4	d_5	d_6
	Antony and Cleopatra	**Julius Caesar**	**The Tempest**	**Hamlet**	**Othello**	**Macbeth**
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

- $w_{Antony,d_1} = \frac{tf_{Antony,d_1}}{\max\limits_{t_i \in d_1} tf_{t_i,d_1}} \cdot \log\left(\frac{N}{df_{Antony}}\right) = \frac{157}{232} \cdot \log\left(\frac{6}{2}\right) = 0.32$

tf-idf weighting

- Increases with the number of occurrences within a document
  - Common in doc → high tf → high weight
- Increases with the rarity of the term in the collection
  - Rare in collection → high idf → high weight