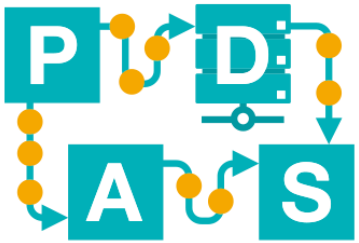
Text Mining - Instruction

Lecture 16

IDS-L16-I



Chair of Process and Data Science

RWTH AACHEN UNIVERSITY

Short summary of the lecture

- **Basics and definitions of Text Mining.**
- **Text preprocessing: tokenization, stopword removal, stemming and lemmatization.**
- **The BoW model and some applications.**
- **Tfidf weighting.**

Definitions - Recap

- **Corpus**: collection of pieces of text with a consistent nature (articles, forum posts, tweets, etc)
- **Documents**: the fragments of text in a corpus
- **Annotated corpus**: corpus in which the documents or fractions of documents have been enriched with metadata

Preprocessing - Recap

Necessary to preprocess text to get structured data.

1. **Dividing the text in discrete units (tokenization)**
2. **Removing irrelevant tokens (stopwords)**
3. **Normalize the tokens so that the same concept is always represented by the same token**
    1. **Represent concepts with stems**
    2. **Represent concepts with lemmas**

Bag of Words - Recap

You can use the Bag of Words model (BoW) to represent documents in a corpus.

It is simply the bag or multiset of the tokens contained in a document of the corpus.

Advantages:

- Simple
- Effective in some applications

Disadvantages:

- The order of the words in a document is lost
- The representation is very sparse (one feature per word in the dictionary)

Tfidf - Recap

Tfidf weighting is a simple way to represent the relevance of a word in a document.

$$tf(w,d) = \#of\ occurrences\ of\ word\ w\ in\ document\ d$$

$$idf(w) = log_2\left(\frac{N}{\#of\ documents\ that\ contain\ w\ at\ least\ once}\right)$$

$$tfidf(w,d) = tf(w,d) * idf(w)$$

Tfidf - Recap

Even if extremely simple, many querying systems rely on (variations of) tf-idf!

- Given a query and a corpus
- For each document in the corpus
  - Compute $score(query, d) = \sum_{w \in query} tfidf(w, d)$
- Rank documents by score
- Return first n documents

Querying systems - exercise

D1: ‘Cats are the only pet of the felines family, while dogs are canids.’

D2: ‘Cats are the third-most popular pet in the US.’

D3: ‘Dogs have been selected for millennia as pet animals.’

D4: ‘Normally, dogs are not aggressive towards other dogs outside their territory.’

This is the example corpus shown in the lecture.

Can you find the ranking of the four documents in the corpus for these three queries?

Q1: ‘dog’

Q2: ‘dog pet’

Q3: ‘dog cat’

Assume that plural normalizes on singular. Stopword removal not necessary.

Querying systems - solutions

	D1_tf	D1_idf	D1_tfidf
dog	1	0.415037	0.415037
pet	1	0.415037	0.415037
cat	1	1	1

	D2_tf	D2_idf	D2_tfidf
dog	0	0.415037	0
pet	1	0.415037	0.415037
cat	1	1	1

	D3_tf	D3_idf	D3_tfidf
dog	1	0.415037	0.415037
pet	1	0.415037	0.415037
cat	0	1	0

	D4_tf	D4_idf	D4_tfidf
dog	2	0.415037	0.830075
pet	0	0.415037	0
cat	0	1	0

	D1	D2	D3	D4
Q1	0.415037	0	0.415037	0.830075
Q2	0.830075	0.415037	0.830075	0.830075
Q3	1.415037	1	0.415037	0.830075

Tfidf: some observations

As you may have noticed, there is an obvious problem with the "standard" version of tdfidf.

Suppose that J. Random Hacker, a malevolent user, manages to access our database.

Tfidf: some observations

He adds a document to the corpus.

“THIS TEXT PROPAGATES A HORRIBLE VIRUS if your read this it’s already too late dog dog dog dog dog dog dog dog.”

You can probably imagine where this is going. All the users ingenuously looking for dogs-related content will be infected.

Tfidf: some observations

This actually happened with search engines in the old days, and was quite a big problem!

People use to hide long sequences of words unrelated to the page in small font and with the same color of the background, in order to appear in search engine query results.

Today search engines have gotten smarter, and you need more sophisticate techniques to fool page rankings.

Tfidf: some observations

The tf term in tfidf weighting is usually way too strong.

With the formulation you have seen, the tfidf scoring assumes that a document containing 10 occurrences of the word ‘dog’ is 10 times more relevant than a document containing 1 occurrence, and this is almost always not the case.

A possible solution is taking the logarithm of the term frequency.

$$tfidf(w,d) = log(tf(w,d)) * idf(w)$$