Keywords

HAP/LAP. Corpus Linguistics.

Keyword Extraction

- Locate and define words that describe document topics.
- Useful in many areas
    - Search engines
    - Document information
    - Automatic summarization
- Challenging problem due to non regular nature of language.
- Example:

  http://www.cortical.io/demos.html

Keyword Extraction

- To extract keywords we need more than one corpus
  - Corpus A: the corpus we want to analyze.
  - Corpus B: a reference corpus to compare against.
- Many techniques, based on frequency analysis.

Keyword Extraction: tf-idf

- Basic technique, but gets very good results.
- In principle, frequent terms are candidates to be keywords
  - but some terms are always very frequent (“the”, “of”, ...)
- Idea: analyze frequency of terms within document and across documents.
  - (+) term appears frequently in the document.
  - (-) term appears frequently in all the documents.
- Ideally, we want terms that appear frequently in one document but do not appear in other documents.

Keyword Extraction: tf

- **tf**: term frequency

  $\mathrm{tf}(t, d) = f_d(t)$, frequency of term t in document d
- There are other choices:

$$\mathrm{tf}(t, d) = \begin{cases} \log(f_d(t) + 1) & t \in d \\ 0 & otherwise \end{cases}$$

$$\mathrm{tf}(t, d) = \frac{1}{2} + \frac{\frac{1}{2} f_d(t)}{\max\{f_d(w) : w \in d\}}$$

- The last formula tries to minimize the fact that longer documents have higher frequencies.

Keyword Extraction: idf

- **idf**: inverse term frequency: how much information provided by the term.

  $\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D \ : \ t \in d\}|}$

  where

  t the term
  D set of all documents
  N number of documents in D
  $| \{d \in D \ : \ t \in d\} |$ number of documents that contain term t

Keyword Extraction: tf-idf

- Puting tf and idf together:
  $\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$

Keyword Extraction: BM25

- Alternative to tf/idf
- $\mathrm{tf}(t,d) = \frac{f_d(t)\cdot(k_1+1)}{f_d(t)+k_1\cdot\left(1-b+b\cdot\frac{|d|}{\text{avgdl}}\right)}$

  where

  $|d|$ length of document.
  avdgl: average length of documents.
  k_1: free parameters (usually $k_1 \in [1.2, 2.0]$)
  b (usually $b = 0.75$)
- $\mathrm{idf}(t,D) = \log\frac{N-n(t)+0.5}{n(t)+0.5}$

  N size of D
  $n(t)$ number of documents containing t
- More information: https://labur.eus/sc069

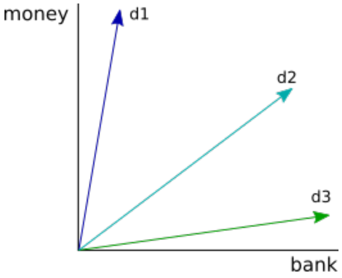
Keyword Extraction: hands on

Exercise

- Using `df.py`, create `word.df` file with following format:
  Number_of_documents
  word TAB number_of_documents_where_the_word_appears
- Complete `tfidf.py` script that given `word.df` and a `.tab` file outputs the tfidf values of words:
  word TAB tfidf
- Note: use the formula $\mathrm{tfidf} = \log(\mathrm{tf} + 1) \times \mathrm{idf}$

Vector Space model

- Represent documents as vectors
- Dimensions are vocabulary words
- Document similarity is proportional to the angle between vectors:
  - parallel: documents are the same
  - perpendicular: documents are completely different



G. Salton , A. Wong , C. S. Yang, "A vector space model for automatic indexing, Communications of the ACM", v.18 n.11, p.613-620, Nov. 1975

Using keywords to compare documents

Idea: use keywords to compare documents against.

- Create a vector for each document.
  - topK keywords according to tf-idf

    $d_1 = \mathbf{u} = (\mathrm{w}_1 : \mathrm{tfidf}(w_1, d_1, D), \mathrm{w}_2 : \mathrm{tfidf}(w_2, d_1, D), \ldots)$

    $d_2 = \mathbf{v} = (\mathrm{w}_4 : \mathrm{tfidf}(w_4, d_2, D), \mathrm{w}_{10} : \mathrm{tfidf}(w_{10}, d_2, D), \ldots)$

Document	whale	bride	widow	bullet	...
Moby Dick	10.1	2.0	1.4	0.9	...
Emma	0.0	3.4	0.9	0.0	...
Father Brown	0.0	0.0	0.5	5.1	...
...	...	...	...	...	...

Keyword extraction: hands on

Exercise

- Check `docSimilarity.py` to obtain similarity scores of two documents.

$$\cos(d_j, q) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{\|\mathbf{d_j}\| \|\mathbf{q}\|} = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}}$$

Document similarity: hands on

Exercise

- Obtain similarity among documents using list of terms weighted by tf-idf. Use `create_tfidf.sh` script for creating all `tfidf` vectors, and the `docsim.csv` file.
- Open `docsim.csv` in libreoffice calc and compare documents.