Document Classification: Part I

Statistical Analysis and Document Mining

Spring 2021

Vasilii Feofanov

Credits: Massih-Reza Amini

Université Grenoble Alpes
vasilii.feofanov@univ-grenoble-alpes.fr

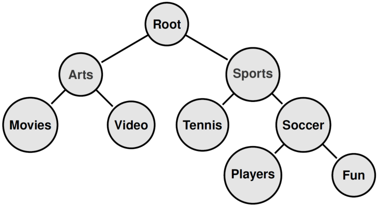
Outline

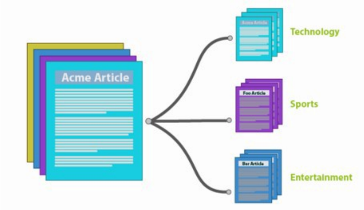
Introduction

- Last two weeks, we dealt with classification in the general case. Now, we look particularly at the class of text applications, which arises additional questions on how we preprocess text, store data and learn the model.

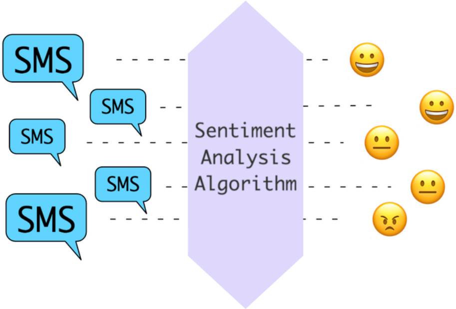
- Last two weeks, we dealt with classification in the general case. Now, we look particularly at the class of text applications, which arises additional questions on how we preprocess text, store data and learn the model.
- In many scenarios, we observe text data:
  Spam detection: given a letter, recognize it is a spam or not.

- Last two weeks, we dealt with classification in the general case. Now, we look particularly at the class of text applications, which arises additional questions on how we preprocess text, store data and learn the model.
- In many scenarios, we observe text data:
  Genre classification: automatically classify articles by genre.





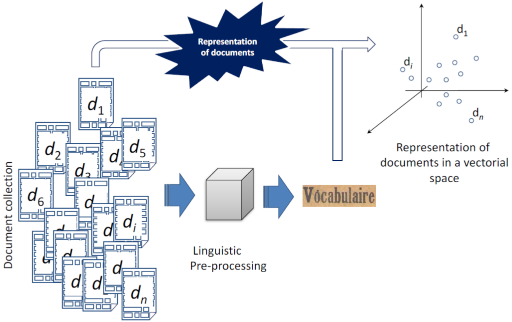
- Last two weeks, we dealt with classification in the general case. Now, we look particularly at the class of text applications, which arises additional questions on how we preprocess text, store data and learn the model.
- In many scenarios, we observe text data:
  Sentiment analysis: determine emotions of the writer.



Introduction

- Last two weeks, we dealt with classification in the general case. Now, we look particularly at the class of text applications, which arises additional questions on how we preprocess text, store data and learn the model.
- In many scenarios, we observe text data:
  Language identification: recognize the language of the text.

- Our observations are labeled raw documents. Before learning, we have to process text first to get "numbers".
- By linguistic preprocessing we distinguish unique words (*terms*) that form our *vocabulary*.
- Using some rule, we represent data in a feature space. Usually, each feature is a word with values indicating its importance.



Outline

Segmentation

- *Segmentation* (tokenization): separate a sequence of characters into semantic elements, or *words*.
- *Term* (type of words): class of all words having the same sequence of characters.
- *Example*:
  "The cat sat on the mat."
  Words: The, cat, sat, on, the, mat
  Terms: the, cat, sat, on, mat
- *Dificulty*: Tokenization is language specific.

Example: Segmentation In French

In French, the following issues may arise during the segmentation process:

- Lexical components with hyphens:
  chassé-croisé, peut-être, rendez-vous
- Lexical components with an apostrophe:
  jusqu'où, aujourd'hui, prud'homme
- Idiomatic expressions:
  au fait, poser un lapin, tomber dans les pommes
- Contracted forms:
  j', M'sieur, Gad'zarts (les gars des Arts et Métiers)
- Acronyms:
  K7, A.R., CV, càd, P.-V.

Normalization

1. *Textual normalization*: consists in reducing the words of a same family to their canonical forms.
   - Punctuation: suppression of points and hyphens;
   - Lower-upper case: transform all upper cases to lower cases;
   - Accents: suppression of accents.
2. *Linguistic normalization* consists in
   - Rooting: replace each word by its root;
   - Stemming: replace each word by its canonical form.

Examples of Preprocessing

Non-spam message before prepocessing

Subject: Re: 5.1344 Native speaker intuitions The discussion on native speaker

intuitions has been extremely interesting, but I worry that my brief intervention may have muddied the waters. I take it that there are a number of separable issues. The first is the extent to which a native speaker is likely to judge a lexical string as grammatical or ungrammatical per se. The second is concerned with the relationships between syntax and interpretation (although even here the distinction may not be entirely clear cut).

Examples of Preprocessing

Non-spam message before prepocessing

Subject: Re: 5.1344 Native speaker intuitions The discussion on native speaker

intuitions has been extremely interesting, but I worry that my brief intervention may have muddied the waters. I take it that there are a number of separable issues. The first is the extent to which a native speaker is likely to judge a lexical string as grammatical or ungrammatical per se. The second is concerned with the relationships between syntax and interpretation (although even here the distinction may not be entirely clear cut).

Non-spam message after prepocessing

re native speaker intuition discussion native speaker intuition extremely interest worry brief intervention muddy waters number separable issue first extent native speaker likely judge lexical string grammatical ungrammatical per se second concern relationship between syntax interpretation although even here distinction entirely clear cut

Preprocessed Documents from Different Classes

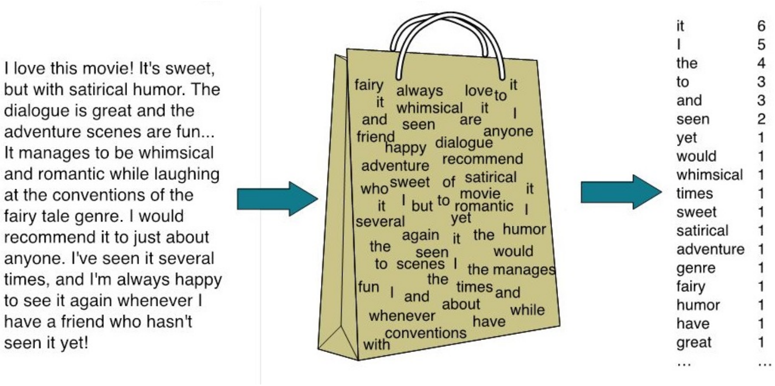
Non-spam message after prepocessing

re native speaker intuition discussion native speaker intuition extremely interest worry brief intervention muddy waters number separable issue first extent native speaker likely judge lexical string grammatical ungrammatical per se second concern relationship between syntax interpretation although even here distinction entirely clear cut

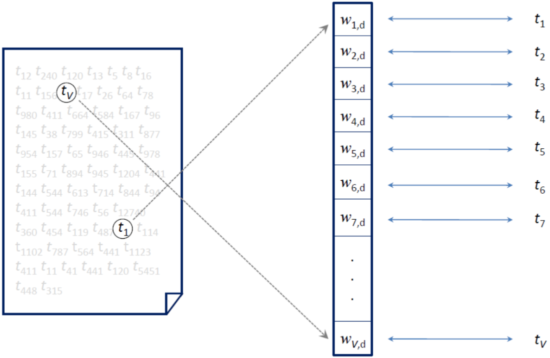
Spam message after prepocessing

financial freedom follow financial freedom work ethic extraordinary desire **earn** least per month work home special skills experience required train personal support need ensure success legitimate homebased **income opportunity** put back control finance life ve try opportunity past fail live **promise**

- The idea of the bag-of-words is to take simply into account only the word appearances ignoring the word order in a sentence (like if we put all words in a bag).



- Given a document, we assign to each term t_j a specific value $w_{j,d}$. Then, a document is represented as a vector: $\mathbf{d} = (w_{j,d})_{j=1}^{V}$.
- How we define importance $w_{j,d},\ j \in \{1, \ldots, V\}$?



Binary and Term Frequency (TF) Rules

- *Binary weighting rule:* if term t_j is present in the document, then $w_{j,d} = 1$. Otherwise, $w_{j,d} = 0$.

Binary and Term Frequency (TF) Rules

- *Binary weighting rule:* if term t_j is present in the document, then $w_{j,d} = 1$. Otherwise, $w_{j,d} = 0$.
  - What is the main drawback of this approach?

Binary and Term Frequency (TF) Rules

- *Binary weighting rule:* if term t_j is present in the document, then $w_{j,d} = 1$. Otherwise, $w_{j,d} = 0$.
  - What is the main drawback of this approach?
- *TF weighting rule:* assuming that more frequent terms are more important, for each t_j, we compute the *term frequency*, i.e. the number of occurrences of t_j in the document: $w_{j,d} = \mathrm{tf}_{t_j,d}$.

Binary and Term Frequency (TF) Rules

- *Binary weighting rule:* if term t_j is present in the document, then $w_{j,d} = 1$. Otherwise, $w_{j,d} = 0$.
  - What is the main drawback of this approach?
- *TF weighting rule:* assuming that more frequent terms are more important, for each t_j, we compute the *term frequency*, i.e. the number of occurrences of t_j in the document: $w_{j,d} = \mathrm{tf}_{t_j,d}$.
  - Let N_d be the number of words in the document. Then, $\sum_{j=1}^{V} \mathrm{tf}_{t_j,d} = N_d$.

Binary and Term Frequency (TF) Rules

- *Binary weighting rule:* if term t_j is present in the document, then $w_{j,d} = 1$. Otherwise, $w_{j,d} = 0$.
  - What is the main drawback of this approach?
- *TF weighting rule:* assuming that more frequent terms are more important, for each t_j, we compute the *term frequency*, i.e. the number of occurrences of t_j in the document: $w_{j,d} = \mathrm{tf}_{t_j,d}$.
  - Let N_d be the number of words in the document. Then, $\sum_{j=1}^{V} \mathrm{tf}_{t_j,d} = N_d$.
  - What is the main drawback of this approach?

Data Storage

- In many applications, there is access to a large collection (n) of documents.

Data Storage

- In many applications, there is access to a large collection (n) of documents.
- This situation necessarily leads to the large vocabulary size V as well.

Data Storage

- In many applications, there is access to a large collection (n) of documents.
- This situation necessarily leads to the large vocabulary size V as well.
- *Problem:* Need to store a matrix of size $n \cdot V$.

Data Storage

- In many applications, there is access to a large collection (n) of documents.
- This situation necessarily leads to the large vocabulary size V as well.
- *Problem:* Need to store a matrix of size $n \cdot V$.
  - What is approximately the size of a data set in GB, if $n = 200,000$, $V = 10,000$ and one entry needs 8 bytes?

Data Storage

- In many applications, there is access to a large collection (n) of documents.
- This situation necessarily leads to the large vocabulary size V as well.
- *Problem:* Need to store a matrix of size $n \cdot V$.
  - What is approximately the size of a data set in GB, if $n = 200,000$, $V = 10,000$ and one entry needs 8 bytes?
  - Around 16 GB!

French Wikipedia Collection: Statistics

Variables	Values
# of documents in the collection	**1,349,539**
Total # of occurrences of words	696,668,157
Average # of words per document	416
Size of the pre-processed collection on the disk	4.6 GB
Total # of types of words	757,476
Total # of types of words after rooting	604,244
Size of the vocabulary	**604,244**
Average # of terms per document	**225**
Size of the collection after removing a stop-list	2.8 GB

Sparse Matrices

- From the statistics, we can see that the number of features in the bag-of-words will be $604,244$, but, in average, only 225 of them will not be equal to 0 for each document.
- Hence, we deal with *sparse matrices*: most of entries are zeros.
- To reduce the storage overhead, we can store data in a sparse format by keeping non-zero elements only.

The Compressed Sparse Row (CSR) Format

The *CSR format* stores a matrix $(n \times V)$, where NNZ entries are not zero, as 3 one-dimensional arrays Val, CI, RI.

- Val stores all non-zero entries.
- CI stores their column indices, so the size of CI is also NNZ.
- RI stores cumulatively the number of non-zero entries per row. The size of RI is $n + 1$. $RI[1] = 0$, $RI[n + 1] = NNZ$. To get the number of non-zero entries for row i, we compute $RI[i + 1] - RI[i]$.

LibSVM Format

- In machine learning, it is popular to store a data set in the *LibSVM format*.
- The data is stored as a 2d array, in which rows may have different number of columns.
- For each row, the first element is the class label, and the rest are *column-index:value* pairs that correspond to non-zero entries.

y	index-value		index-value
2	5:0.356	...	9:1000
3	2:10.2	...	15:0.01

French Wikipedia Collection: Statistics

Variables	Values
# of documents in the collection	1,349,539
Total # of occurrences of words	696,668,157
Average # of distinct words per document	416
Size of the pre-processed collection on the disk	4.6 GB
Total # of types of words	**757,476**
Total # of types of words after rooting	604,244
Size of the vocabulary	604,244
Average # of terms per document	225
Size of the collection after removing a stop-list	2.8 GB

Two Laws of IR

- The encyclopedia "Grand Robert" contains around 75,000 words. The most extensive record shows that the French language would contain about 700,000 words. Why in French Wikipedia we found even more words (757,476)?

Two Laws of IR

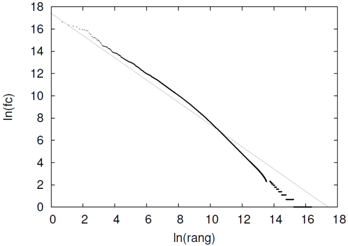
- The encyclopedia "Grand Robert" contains around 75,000 words. The most extensive record shows that the French language would contain about 700,000 words. Why in French Wikipedia we found even more words (757,476)?

- When the collection was filtered by removing a stop-list ("a", "the", "of", etc.) of size 200 words, the average number of terms was reduced in documents from 416 to 225 (around 45% reduction). Why?
  In addition, their filtering reduces the space on the disk of about 39% (from 4.6 GB to 2.8 GB).

The number of occurrences $fc(word)$ of a word $word$ in a document collection is inversely proportional to its rank:

$$\forall word: \; fc(word) \approx \frac{\lambda}{\mathrm{rang}(word)}.$$

$\Rightarrow$ The k-th most frequent word is approximately k times less present than the most frequent one.



Rank	Word	Freq.	%
1	the	22,038,615	4.9%
2	be	12,545,825	2.79%
3	and	10,741,073	2.39%
4	of	10,343,885	2.3%
5	a	10,144,200	2.25%
6	in	6,996,437	1.56%
7	to (i.m.)	6,332,195	1.41%
8	have	4,303,955	0.96%
9	to (p.)	3,856,916	0.86%
10	it	3,872,477	0.86%

Top 10 frequent word from the 450 million word corpus (`https://www.wordfrequency.info`).

Filtering

- We suppress very frequent words that are present in most of the documents and do not bring any information.

Filtering

- We suppress very frequent words that are present in most of the documents and do not bring any information.
- *Example*:

  "The cat sat on the mat."
  Before filtering: the, cat, sit, on, mat
  After filtering: cat, sit, mat

Filtering

- We suppress very frequent words that are present in most of the documents and do not bring any information.
- *Example*:

  "The cat sat on the mat."
  Before filtering: the, cat, sit, on, mat
  After filtering: cat, sit, mat
- How many frequent words we should suppress?

Filtering

- We suppress very frequent words that are present in most of the documents and do not bring any information.
- *Example*:

  "The cat sat on the mat."
  Before filtering: the, cat, sit, on, mat
  After filtering: cat, sit, mat

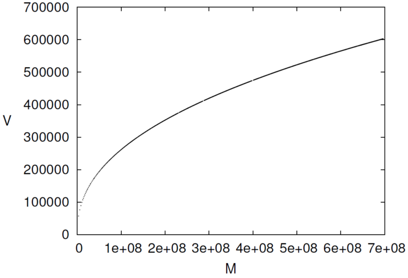
- How many frequent words we should suppress?
  - We should be aware that removing of too many words may harm prediction performance. Usually, modern packages have tools to remove stop-list words of most spoken languages.

The size of the vocabulary V increases sub-linearly with respect to the total number of words M present in a collection:

$$V = k \cdot M^{\beta},$$

where k and β are parameters that are dependent on the collection. Typically, in English text corpora $k \in [10, 100]$, and $\beta \in [0.4, 0.6]$.

$\Rightarrow$ Larger the collection size, larger the vocabulary size.



Term Frequency vs Document Frequency

- When we filter the stop-list words, the term frequency weighting rule might work better.

Term Frequency vs Document Frequency

- When we filter the stop-list words, the term frequency weighting rule might work better.
- However, less frequent terms can also have large importance, since they can be class-specific.

  Example: Medical Prescription vs Recipe.

take	water	glass	eat	wait	...	paracetamol	sugar	stomach
7	6	4	4	4	...	0	2	0
6	7	4	3	5	...	1	0	1

Term Frequency vs Document Frequency

- When we filter the stop-list words, the term frequency weighting rule might work better.

- However, less frequent terms can also have large importance, since they can be class-specific.

  Example: Medical Prescription vs Recipe.

take	water	glass	eat	wait	...	paracetamol	sugar	stomach
7	6	4	4	4	...	0	2	0
6	7	4	3	5	...	1	0	1

- We want to take into account document frequency of terms by diminishing the weight of terms that occur frequently across documents and increasing the weight of terms that occur rarely in average.

TF-IDF Weighting Rule

The TF-IDF rule is a trade-off between term frequency and document frequency:

- Normalized term frequency (tf part):

$$\frac{\mathrm{tf}_{t_j,d}}{\sum_{j=1}^{V} \mathrm{tf}_{t_j,d}} = \frac{\mathrm{tf}_{t_j,d}}{N_d}.$$

TF-IDF Weighting Rule

The TF-IDF rule is a trade-off between term frequency and document frequency:

- Normalized term frequency (tf part):

$$\frac{\mathrm{tf}_{t_j,d}}{\sum_{j=1}^{V} \mathrm{tf}_{t_j,d}} = \frac{\mathrm{tf}_{t_j,d}}{N_d}.$$

- Inverse document frequency (idf part):

$$\ln \frac{n}{\mathrm{df}_{t_j}} := \ln \frac{n}{\sum_{i=1}^{n} \mathbb{I}(\mathrm{tf}_{t_j,d_i} \neq 0)}.$$

TF-IDF Weighting Rule

The TF-IDF rule is a trade-off between term frequency and document frequency:

- Normalized term frequency (tf part):

$$\frac{\mathrm{tf}_{t_j,d}}{\sum_{j=1}^{V} \mathrm{tf}_{t_j,d}} = \frac{\mathrm{tf}_{t_j,d}}{N_d}.$$

- Inverse document frequency (idf part):

$$\ln \frac{n}{\mathrm{df}_{t_j}} := \ln \frac{n}{\sum_{i=1}^{n} \mathbb{I}(\mathrm{tf}_{t_j,d_i} \neq 0)}.$$

- Then, the tf-idf weight is defined as:

$$w_{t_j,d} = \frac{\mathrm{tf}_{t_j,d}}{N_d} \ln \frac{n}{\mathrm{df}_{t_j}}.$$