What feature extraction methods are available for text classification?

Because a machine learning model needs numbers to learn, the text needs to be converted to numbers. After preprocessing, this is the second step in the pipeline for text classification. We call this translation from text into numbers ‘vectorization’ or ‘feature extraction’. This vectorization step is unique to text classification. As expected, different types of vectorization can be used. Depending on the model and context, a certain method can be picked. The following paragraphs describe different vectorization techniques.

Bag of Words

The simplest and easiest to understand method is ‘Bag of Words’. This method disregards grammar and word order, but keeps multiplicity.[1] It created an array with the length of all unique words in the entire dataset for each document. In this array, it simply counts how much every word occurs in that document and increases the count by 1. An example of this can be seen in figure 1. In the third document, the word ‘account’ appeared twice and in the fifth and sixth document, the word ‘account’ appeared once.

	aaa	abfubinaca	abl	abpinaca	acc	accent	accep	accept	access	account	...	ye	year	yet	yo	york	your	youtub	yr	zealand	zip
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	2	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0

Tf-idf

Tf-idf is short for ‘term frequency–inverse document frequency’. It takes into account not only how often a word appears, but also how often it occurs in other documents. The tf-idf increases proportionally to the number of times a word appears in a document and is offset by the number of documents in the corpus that contain the word. This helps to counter for the fact that a lot of words may appear in a lot of documents, which decreases the meaning of the word for a specific document.

The term frequency is the amount of times a word appears in a single document. The inverse document frequency decreases the weight of words that appear more frequently among the documents and increases the weight of those that occur rarely. A high weight in tf-idf is reached when a word has a high document frequency and a low inverse document

[1] https://en.wikipedia.org/wiki/Bag-of-words_model

```
# 'Chemicals':
  . Most correlated unigrams:
. camphor
. phosphorus
  . Most correlated bigrams:
. selected with
. orders selected
```

frequency. In other words: when a word occurs a lot in one document, but very rarely in others, it is highly significant for that document and thus gets a higher weight.

Tf-idf can also create n-grams, which look at which words are mostly appear together. figure 2 shows a snippet of bigrams we created using tf-idf.

Tf–idf is one of the most popular term-weighting schemes today; 83% of text-based recommender systems in digital libraries use tf–idf.[2] For our data, it seemed to work best, so this was our method of choice as well.

Word2Vec

Word2Vec is a vectorization method that is more complex than the others. It has a sense of context of the words which means that similar words will appear close together in the vector space. For example, the words ‘Amsterdam’ and ‘Delft’ will appear close together, even though the characters in the words are very different. Word2Vec knows that they’re both cities and thus are similar type of words.[3]

Doc2Vec

As the name suggests, Doc2Vec works in a similar way as Word2Vec. It also takes into account the meaning of words, but adds a paragraph id. This combined document vector is intended to represent the concept of a document.[4]

FastText

FastText is Facebook’s machine learning library that uses neural networks to create vectors. The model allows for unsupervised and supervised learning and Facebook makes pretrained models available for 294 languages.[5]

[2] https://en.wikipedia.org/wiki/Tf–idf
[3] https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e
[4] https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e
[5] https://en.wikipedia.org/wiki/FastText