Text Analytics

Machine learning algorithms are developed for numeric data. However, in the era of social media, there is so much text generated which needs to be analyzed especially for obtaining consumer insights. In addition to social media, there are significant number of text analytics applications in healthcare, publications and technology companies.

The process of converting text data to numeric data is called feature extraction. Let's look at an example below where text from 4 documents are converted to numeric features based on the count of words

	blue	bright	can	see	shining	sky	sun	today
Doc1	1	0	0	0	0	1	0	0
Doc2	0	1	0	0	0	0	1	1
Doc3	0	1	0	0	0	1	1	0
Doc4	0	1	1	1	1	0	2	0

Term Frequency

The above matrix is called Term Frequency matrix where term frequency $tf_{t,d}$ of term t in document d is defined as the number of times that t occurs in d. Let's take another example where documents are novels of Shakespeare.

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

Document Frequency

Rare terms are more informative than frequent terms. Consider a term that is rare (e.g.,Calpurnia). A document containing this term is very likely to be relevant to the query Calpurnia. We want a high weight for rare terms like Calpurnia. We will use document frequency (df) to capture this.

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. $df_t \leq N$ where N is the number of documents

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

$$\text{idf}_t = \log_{10}(N/\text{df}_t)$$

- *We use log (N/df_t) instead of N/df_t to "dampen" the effect of idf.*

IDF provides high values for rare words and low values for common words.

For a collection of 10000 documents (N = 10000)

$$\log\left(\frac{10000}{10000}\right) = 0$$

$$\log\left(\frac{10000}{5000}\right) = 0.301$$

$$\log\left(\frac{10000}{20}\right) = 2.698$$

$$\log\left(\frac{10000}{1}\right) = 4$$

Tf-Idf Weighting

Many a times, both term frequency and Inverse Document frequency are considered together while extracting features.

$$\text{tf.idf}_{t,d} = \log(1+\text{tf}_{t,d}) \times \log_{10}(N/\text{df}_t)$$

It increases with the number of occurrences within a document. It also increases with the rarity of the term in the collection. Given a search query term, the document are ranked for relevance based on the Score

$$\text{Score}(q,d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

An illustrative tf-idf feature is shown below

	Antony	Brutus	Caesar	Calpurnia	Cleopatra	mercy	worser
Antony and Cleopatra	5.25	1.21	8.59	0	2.85	1.51	1.37
Julius Caesar	3.18	6.1	2.54	1.54	0	0	0
The Tempest	0	0	0	0	0	1.9	0.11
Hamlet	0	1	1.51	0	0	0.12	4.15
Othello	0	0	0.25	0	0	5.25	0.25
Macbeth	0.35	0	0	0	0	0.88	1.95

Text Analytics Steps

- Do cleanup of text if needed
- Extract features from text documents/tweets (e.g: tfidf) and build a numeric matrix
  - *In practical cases there would be several 1000s of terms. Saving them in a regular matrix format will occupy large memory space*
  - *1000 documents with total of 100,000 terms with each tf.idf value saved as 32bit float when saved as raw matrix will occupy 4GB RAM*
  - *Most of the document x term cells will be 0. Hence better saved as sparse matrix*
- Build clustering or classification models on the numeric data