Informatics Lecture Notes

Multimedia Information Systems

Module 2: Statistical Text Analysis (I)

Lecturer: Dr. Manuel Pita
Tutor: João Correia

Autumn 2017

Contents

1 Introduction to Statistical Text Analysis

In this module we focus on the most widespread and present media form: *text.* Text is everywhere, in spoken language, books, websites, and so on. It is no surprise that the success of big search engines is based on the appropriate analysis of text. Deriving 'meaning' from text is far from trivial, indeed it is a very difficult task. Consider that meaning in text is created by distributions of words in specific language, following very specific and diverse grammar rules. These arrangements are further nuanced by cultural codes, shorthands, metaphors, analogies, irony, specific references and so on.

2 The Bag-Of-Words Pipeline

The life cycle statistical text processing has four stages at its core. They are illustrated in Figure 1. The starting point of this process is the availability of a **corpus** which is nothing else than a collection of text documents. For example, we can have a corpus on political debates, or cooking recipes, the news broadcast by a given agency in a time period, and so on. Corpora (plural of corpus) thus often gather many documents on a given theme, and our goal is to find the different topics that make up that theme.

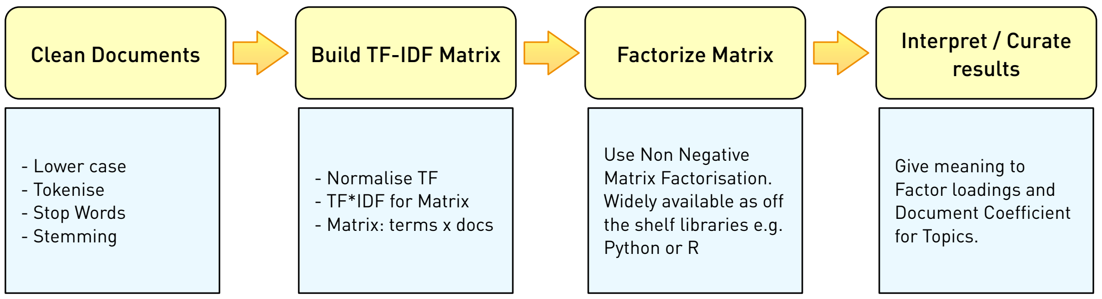
Clean Documents
Build TF-IDF Matrix
Factorize Matrix
Interpret / Curate results
- Lower case
- Tokenise
- Stop Words
- Stemming
- Normalise TF
- TF*IDF for Matrix
- Matrix: terms x docs
Use Non Negative Matrix Factorisation. Widely available as off the shelf libraries e.g. Python or R
Give meaning to Factor loadings and Document Coefficient for Topics.

Figure 1: The four main stages in the life cycle of Statistical Text Processing. In this document we cover the first two stages.

Once we have our corpus, the first stage is concerned with **cleaning the documents.** The main idea here is to remove 'noise' in the form of text that has little to say about what a document is speaking about. Statistical text analysis (STA) is concerned with the distributions of words. This is so in the sense that a document that contains e.g. many copies of the words 'oven', 'cook' and 'onion' are likely to be about recipes/food. STA is not concerned with grammar rules of any kind. For this reason, all connectors, punctuation marks and so on are not important to STA (and indeed they are eliminated as we will see later). Other methods for analysing text in the field of Deep Natural Language processing are interested in (and use) grammar rules but we don not study Deep NLP in this course.

3 Cleaning the Documents in the Corpus

To clean the documents in our corpus we iteratively load each document into our computer program, changing the **(1) text to lower case**. Sometimes (esp. when using dictionaries along the analysis pipeline) we can choose to keep all proper capitalised, but in general text is all changed to standard lower case. Then we **(2) tokenise** the text which turns the document's string into a list of atoms. Each such atom is a word, or a punctuation mark. The next step is to **(3) remove stop words** effectively deleting all punctuation as well as all terms that appear with high probability in the document's source language. For example in English terms such as 'in', 'at', 'to' and so on, are all eliminated because they do not give us information about what the document is about. This already reduces significantly the number of terms that can appear in the documents of our corpus. Notice that the all these terms still appear the same number of times, but each time as the lemma 'map'.

After these steps, each document has been converted to a list of terms. The cleaning stage is not yet completed however. A following step is taken to represent all the different forms of a single concept using only one token. This step is called **stemming**, and basically it is concerned with changing all word variants into their base 'Lemma'. In this way, terms such as mapped, mapping, mapper and maps are replaced by the single lemma 'map'. This clearly compresses the representation of our documents even more.

At this point, and in preparation for the next stage of processing, it is important to derive the **universal** dictionary of terms in the corpus after stemming. This is simply a list of every term that appears at least once in at least one document inside our corpus – that is the reason we call it universal.

Consider the following stemmed corpus consisting of three documents:

$D_1 = \{onion, garlic, garlic, strawberry\}$
$D_2 = \{tomato, butter, tomato, pear, banana\}$
$D_3 = \{milk, flour, onion, onion, tomato\}$

What is the corresponding universal dictionary of terms?

Notice that for implementation of this cleaning stage we can find libraries and resources such as dictionaries of stop words, and lemmatizers for performing stemming available for the most widely used programming languages. In this class we will use Python.

4 Construction of the TF-IDF Matrix

This step is the core of the STA life cycle. What is interesting here is that, while starting with a corpus made of disjoint elements (a collection of documents), we end up with a single corpus representation. This **single representation unifies** the information we have about the comprised documents. Therefore we can use this single data structure to reason about the entire corpus! For this goal, working with an universal dictionary of terms is essential.

4.1 Term Frequency: TF

Term Frequency or simply TF is a numeric quantity used to express the importance of a term inside a document. In its raw form, it is simply the count of times a term appears in a document. However, here we compute TF as a proportion, by dividing this count by the total number of tokens in the (stemmed) document. This means that for example the TF of the term garlic in Document 1 above is $tf(garlic, D_1) = 2/4 = 1/2$ while TF of tomato in Document 2 is $tf(tomato, D_2) = 2/5$.

Computing TF in this way means that the TF value will range between $[0, 1]$ and also that we can compare any pair of documents not having to worry about biases caused by one of the documents being much larger than the other. In other words, this way to compute TF brings all documents to a single standardised numerical place.

4.2 Inverse Document Frequency: IDF

Term frequency does not tell the whole story we need to analyse a corpus. TF relates terms and the documents that contain them. But what about the importance of terms in the corpus? Suppose for example that in our corpus the word 'rice' appears in every document. Suppose that you are the librarian keeping this corpus, and somebody comes searching for a subset of documents in a given topic from your corpus. Imagine this library visitor tells you 'rice'. You go into the box to fetch all the documents that contain that word. Clearly you will come back with the entire box because every document in it contains that term. Was 'rice' a helpful term to support the library visitor's needs? Not really. In fact not helpful at all.

This is where Inverse Document Frequency or IDF comes in handy. This number will represent the importance of a term in a given corpus, calculated using the following formula:

$$IDF_t = Log\left(\frac{N}{df_t}\right)$$

where t refers to the term, N to the total number of documents in the corpus and df_t to the number of documents that contain the term t at least once. Following this we can compute that the IDF of onion is

$$IDF_{onion} = Log\left(\frac{3}{2}\right)$$

What is the IDF of garlic? And does it make any sense to compute the IDF of the term 'plane' in this corpus?

4.3 Term Frequency Inverse Document Frequency: TF.IDF

The final quantity that we will use to measure the importance of a term inside a document that belongs to a corpus os the standard TF.IDF measure which is simply:

$$TF.IDF_t^d = TF_t^d \times IDF_t$$

Here t refers as always to the term, and d to a specific document. What is the effect of multiplying the original normalised TF by the IDF? The IDF acts as a modulator. If the TF is high but the term is everywhere in the corpus, the IDF will be low, so the TF is brought down. If a TF is medium, but the IDF is high then its importance is modulated upwards.

Coming back to the focus of this section now we have all we need to construct our single representation of the corpus as a matrix. This matrix S has rows representing the terms of the universal dictionary for the corpus, and columns representing the contained documents. Therefore a given cell $S_{t,d}$ of the matrix will contain the corresponding $TF.IDF_t^d$.

Compute and validate the entire matrix S for the example corpus in this note in the space below:

Once we have our matrix ready, we can start making sense of what it represents. In other words, we can start doing some clever things with it. It very important to realise that each column of the matrix S represents (stands for) a document d. This means that in the corpus, **d** is still the original source text document, but inside our computer program, it is represented by the column **d** of Matrix S. This column is a distribution of numbers over the space of all words that exist in the corpus. Representing every document across the space of all words in the entire corpus is the biggest value of this matrix. All documents are represented in the same terms. The cells where those numbers are higher correspond to the terms more strongly present in the document that are also informative across the entire corpus. Because documents are effectively represented as vectors it is very easy to compute the angle between two vectors

$$cos(\theta) = \frac{\mathbf{d}_i . \mathbf{d}_j}{\|\mathbf{d}_i\| . \|\mathbf{d}_j\|}$$

Expand this formula below, and compute the cosine of the angles between the possible pairs of documents in the example of this note (hint: there are three possible pairs).

5 Further Notes

The main concepts/ideas you need to understand and explain from this module are the following:

1. What do we mean by bag of words?
2. Why TF alone is not good enough for the matrix representation of a corpus
3. What are the rows in the matrix representation of a corpus?
4. What are the columns in the matrix representation of a corpus?
5. What do we accomplish with the matrix representation of a corpus? Explain in detail
6. What is the use of computing the angle between two documents from a given matrix S
7. What does it mean when we say that IDF is a "modulator"
8. What do we mean by "lemma" when we do word stemming?

The next Module (3) will focus the processing of the matrix S to find topics using Matrix Factorisation.