Text Mining: Extraction of Relevant Expressions and Identification of Relationships between Documents and Keywords

Rúben Carvalho
Faculdade de Ciências e Tecnologia
Faculdade Nova de Lisboa
Email: rg.carvalho@campus.fct.unl.pt

Sofia Morgado
Faculdade de Ciências e Tecnologia
Faculdade Nova de Lisboa
Email: sv.morgado@campus.fct.unl.pt

1. Introduction

Simply put, text mining techniques are used when we want to transform a given set of unstructured text information (set of sentences, paragraphs, documents, books, etc.), into a structured format from which it is possible to identify patterns and find relevant insights. [1]

An important concept within this subject is the Relevant Expressions (RE). As the name suggests, a RE refers to a certain set of words (with the number of words greater than 1), with a very strong meaning, that normally start and finish with a relevant word. A RE represents in condensed form the essential content of a document. In a text, or any other kind of document, only certain sets of words can be considered RE, and they are not necessarily dependent on their frequency in the text for that.

There are several methods for extracting these relevant expressions from sets of text. The main method studied in class was the LocalMax Algorithm. As mentioned by Silva & Lopes (1999) [2], LocalMax is an algorithm that works with a corpus (a text) as an input and produces multi-word units (MWU) from that corpus. The main idea of this algorithm is: from a limited set of words putted together in cohesion groups, a MWU appears as a relevant expression when the cohesion between words (the "glue" value between words, calculated by cohesion measures) stands out in the context of its neighborhood. The higher the "glue" value, the more likely that MWU is to be a relevant expression.

The "glue" value of a given word sequence can be obtained through several cohesion metrics. For all of them, the higher the glue value, the higher the probability that the sequence of words is a relevant expression. However, given the changing frequency values of the n-grams present in the metrics, some may be more sensitive than others, that is, they may present considerably different values for the same set of RE. Although we will not go into a big detail in the description/explanation of each one of them, the metrics we used in this work were: Symmetrical Conditional Probability (SCP), Dice, Mutual Information (MI) and Omega2 (Φ^2).

The concept of a keyword is also important to mention. As Barry & Kogan (2010) [3], a keyword, which is defined as a sequence of one or more words, provide a compact representation of a document content. They are normally the most informative words/set of words on the documents. They can be used to many applications, such as search engines, document classification, document clustering, etc. The keywords can be divided as explicit and implicit. An explicit keyword is word/set of words that occur explicity on the document. It appears on the document that we are studying/analysing. An implicit keyword is a word/set of words that do not occur on the document that we are studying/analysing, but is related to it. It is a word/set of words that it is usually related to the subject that is discussed on that document.

To better select the explicit keywords of our documents, the Tf-Idf (Tf for Text Frequency and Idf for Inverse Document Frequency) algorithm was implemented to order the relevant expressions. This algorithm is a form of encoding that normalizes the token frequency in a document relative to the rest of the corpus, so that the relevance of a token to a document is determined by the scaled frequency of the term's appearance in the document, normalized by the inverse of the term's scaled frequency in the total corpus [4]. This algorithm was also used to select relevant unigrams directly from the text of a set of documents.

1.1. Data set

The data set used in this work was made available by the course teacher, and is called "corpus2mw". This corpus has, approximately 2 million words, from among 3170 documents (corpora). All documents are distinct, however, some may be related in theme and content.

2. Implementation and Results

2.1. Extractor implementation

An extractor of relevant words was implemented and tested in a subset of the corpus which was labeled.

Firstly, the extractor adds spaces before and after each punctuation signal. This step is implemented in the function create_spaces_punctuation(). This is usefull to diminish the number of different tokens in the text, as "Africa" and "Africa," would count as different tokens, although this

should not happen. Also, this way, each punctuation sign will be counted as a token itself, which will help us remove expressions containing punctuation from the list of relevant expressions, as it would not make sense for a relevant expression to be separated by it.

Then, the function create_tokens_dict() is created to extract tokens from corpus. Tokens with the size of one word (unigrams) through seven words (sevengrams) are considered. For this, the function CountVectorizer() from sklearn is used. A dictionary containing each type of tokens and the token count is created.

To add cohesion metrics to these dictionaries, several functions were created, including cohesion_metrics_bigrams() through cohesion_metrics_sevengrams(). They calculate, for each dictionary of tokens and count, the value of SCP, Dice, MI and Omega2. New dictionaries are created with all this information, and are called Bigrams_metrics through Sevengrams_metrics.

These metrics were then used to remove expressions with low values of cohesion. All tokens with the mean of one of the cohesion metrics inferior to the mean of that metric were removed. For this, several steps were implemented. Firstly, the function mean_cohesion() was created to calculate the of a given cohesion metric for a given dictionary, and the function mean_for_all() uses it to calculate the mean of all metrics for a given dictionary. The function min_cohesion() is responsible for removing expressions with a given cohesion metric value inferior to a threshold, and min_cohesion_all() uses it to remove the tokens for all metrics. Finally, mean_for_all() is used to compute the mean of each metric and this value is given to min_cohesion_all() as a threshold.

After this, expressions that include punctuation signs or that are very infrequent, meaning, only appear once in the document, are removed by the function remove_infreq_punct().

Relevant expressions should not begin or end with stop words. So, an algorithm is implemented to remove them. Words tend to have a limited set of ”neighbors”, meaning, words immediately before and after them. For this reason, the number of different neighbors for each word is calculated by splitting unique bigrams. After this, the words are ordered by the number of neighbors they contain and plotted. The plot is shown in Figure 1.

The difference of the number of neighbors is calculated using a step of x words in this plot. The best value for x was found to be 2. This difference will work as a tangent of the curve between those two points. As we want to find the point of maximum deflection, which would separate stop words from the other context words, we calculate the differences between each tangent and find its maximum. The plot showing the differences is shown bellow, in Figure 2.

A list of stop words is therefore created with the words on the right of the word where this maximum happens. (This process was repeated until an optimal number of stop words was selected, given the F1 score). The function remove_stop_words() was created in order to remove

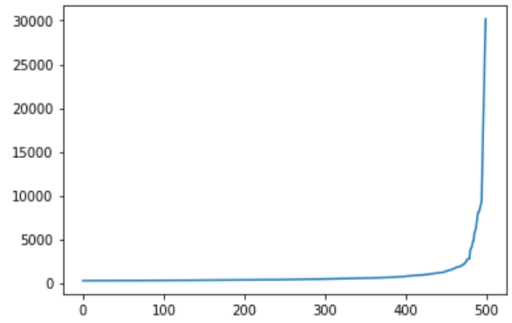

Figure 1. Number of neighbors for the 500 words with more neighbors

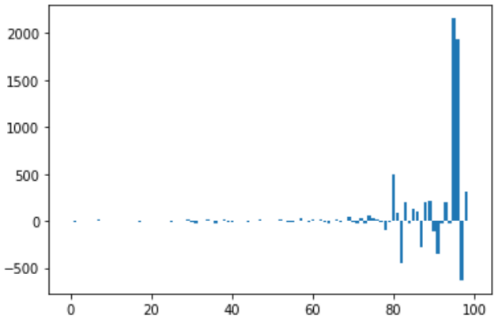

Figure 2. Differences of the tangent of number of neighbors

tokens containing stop words as the first and last word from the relevant expressions dictionaries. A list with the final expressions retrieved as relevant expressions is created.

Finally, the function extractor() is created to receive a corpus and call all the previous functions to retrieve the final relevant expressions from any corpus. This function will later be used in this project.

2.2. Extractor evaluation

To evaluate the extraction of relevant expressions using the extractor, a two labeled corpus was used. These are corpus from which the relevant expressions were removed by hand, composing the groundtruth to which the model will be compared. The function evaluate_extraction() was created to compute the evaluation metrics. Firstly, the number of true positives, false positives, true negatives and false negatives were computed. For comparison, precision, recall and F1 score were used.

One of the corpus was composed of 30 relevant expressions retrieved by hand. Our extractor obtained a value of recall of 0.88, a value of precision of 0.725 and an F1 score of 0.79.

Nevertheless, for a corpus where we can find 208 relevant words, the results, especially for precision, were lower. The values obtained were a value of recall of 0.70, a value of precision of 0.29 and an F1 score of 0.41. These values reflect a closer approximation to the real values of these metrics, as the corpus is larger and the extractor will be used in large corpora.

2.3. Extraction of RE from a complete corpus

The extractor was used to select relevant expressions present in a large corpus composed of several documents. After these relevant words were identified, 5 documents were selected and the RE that were present in each document were added to a dictionary.

In order to find the best 10 RE in each document, the Tf-Idf score was implemented. Then, RE were ordered in descending order of Tf-Idf and the first 10 were selected.

The resulting expressions were considered the most informative RE in each documents and were used to compare documents.

TABLE 1. RE FOR EACH DOCUMENTS 0 AND 1

RE Doc 0	RE Doc 1
comba dão	autonomous province
santa comba	inexhaustible cup
side manhattan	mother of god
york city	province of kosovo
disliked by bruno	autonomous province of kosovo
santa comba dão	orthodox church
19th century	russian orthodox
although there	russian orthodox church
athletic association	
baseball player	

TABLE 2. RE FOR DOCUMENTS 2, 3 AND 4

RE Doc 2	RE Doc 3	RE Doc 4
place overall	doc id	doc id
1st place	org wiki	org wiki
south korea	united states	22 october
won 1st	office of the czech	central coast
1st place overall	21 august	head coach
won 1st place	21 june	
won 1st place overall	czech republic	
band won	laid down	
chicago tribune	most important	
choir won	newly formed	

2.4. Correlation between documents

Through Table 3 we can see that the correlation of the relevant expressions between the 5 selected documents is quite low. There is only a correlation between documents 3 and 4 with a value of 0.2828. This could mean that the documents, for the most part, do not concern the same subject.

2.5. Extraction of relevant unigrams using only Tf-Idf score and unigram lenght

The Tf-Idf score was calculated for each unigram of each of the five documents. This result was multiplied by the length of each word as longer words then to be more informative.

TABLE 3. CORRELATIONS OF RE BETWEEN DOCS

	$Doc0$	$Doc1$	$Doc2$	$Doc3$	$Doc4$
$Doc0$	1	0	0	0	0
$Doc1$	0	1	0	0	0
$Doc2$	0	0	1	0	0
$Doc3$	0	0	0	1	0.2828
$Doc4$	0	0	0	0.2828	1

The values obtained are shown in Tables 4, 5, 6, 7 and 8.

Since longer words are more informative, with a strong semantic meaning, we decided to multiply the length of each word for the value obtained in Tf-Idf, in order to obtain a better and a more meaningful value.

We may compare the relevant expressions extracted using the extractor followed by the application of the Tf-Idf and the unigrams retrieved by using solely the Tf-Idf multiplyed by the length of the unigram. The first method takes longer to compute and produces only n-grams for n equal or superior to 2. On the other hand, these n-grams help contextualize the text better than retrieving only unigrams. For example, "Russian Orthodox Church" is more informative than only "Russian Orthodox", showing that extracting good relevant expressions composed of more words may be more usefull than short n-grams. Nevertheless, we may observe some really informative unigrams, such as "Rothschild" and "Constituency".

TABLE 4. UNIGRAMS FOR THE FIRST DOCUMENT

Unigram	Tf-Idf Value
rothschild	1.71678
constituency	1.66211
guinness	1.37342
stadium	1.20175
matches	1.20175
3745642	1.20175
representation	1.20175
3742301	1.20175
music	1.03882
cockermouth	0.94423

2.6. Finding Implicit Keywords in other documents

A document's implicit keyword is a word that is not contained in the document but is sufficiently semantically related to the document's explicit keywords that it must be significant to the document's context.

TABLE 5. UNIGRAMS FOR THE SECOND DOCUMENT

Unigram	Tf-Idf Value
certification	2.35572
solaris	1.05706
church	1.02339
commentary	0.90605
candidates	0.90605
constituencies	0.84564
network	0.84564
religious	0.81544
amendment	0.81544
administrator	0.78524

TABLE 6. UNIGRAMS FOR THE THIRD DOCUMENT

Unigram	Tf-Idf Value
suffield	1.94498
deciduous	1.75048
tolerates	1.31286
plant	1.21561
shade	0.97249
trowbridge	0.97249
light	0.97249
wiltshire	0.87524
guildford	0.87524
centaurea	0.87524

Four functions were defined in order to calculate the correlations between words. The already extracted Explicit Keywords from 5 documents were compared with relevant expressions from a set of 50 documents. The values of correlation were calculated.

To select the Implicit keywords of the first document, words with a correlation of 0.8 or higher with any of document 1 explicit keywords were selected. The results include: '000 rushing' '000 rushing yards' '19th century' '20 quires' '25 sheets' '450 or 480' '450 or 480 sheets' '480 sheets' 'athletic association' 'autocommit mode' 'cadastral community' 'challenge cup' 'fastest player' 'fastest player to reach' 'film industry' 'gone down' 'language film' 'player to reach' 'property ownership' 'quires and sheets' 'ream of 480' 'ream of 480 sheets' 'rushing yards' 'white stripes'.

As we can see, this category of implicit keywords is quite diverse.

3. Conclusion

A extractor of relevant expressions (RE) form a corpus was implemented. This included finding the most appropriate tokens (from bigrams to sevengrams, where words or punctuation were considered as tokens), remove expressions which only appeared once in the text, removing expressions containing punctuation and stop words. The extractor was evaluated using a labelled corpus.

The RE obtained from a sub-corpus were then used to find the explicit keywords of a set of 5 documents using Tf-Idf. To compare the results, this method was applied also to all the unigrams of the documents, and relevant unigrams were selected after the multiplication of their Tf_Idf with their word length. Finally, the correlation between documents was calculated, and the correlation between explicit keywords of each of the 5 documents and the relevant expressions from other 50 documents was used to find the implicit keywords of each document.

TABLE 7. UNIGRAMS FOR THE FOURTH DOCUMENT

Unigram	Tf-Idf Value
ukrainian	1.29010
polish	1.29010
división	1.14676
argumentation	0.93174
dahlmann	0.86007
leptospermum	0.86007
cooperation	0.78840
tercera	0.75256
railway	0.75256
journalism	0.71672

TABLE 8. UNIGRAMS FOR THE FIFTH DOCUMENT

Unigram	Tf-Idf Value
abramoff	1.83842
minister	1.37881
institution	1.26391
immigration	1.26391
season	1.11242
hayworth	0.91921
expected	0.91921
intercollegiate	0.86176
competition	0.84646
announced	0.83431

References

[1] Education, IBM Cloud. "What Is Text Mining?" What Is Text Mining? — IBM, www.ibm.com, 16 Nov. 2020, https://www.ibm.com/cloud/learn/text-mining.

[2] Ferreira da Silva, J. & Pereira Lopes, G. (1999). A Local Maxima method and a Fair Dispersion Normalization for extracting multi-word units from corpora. FCT/DI. Universidade Nova de Lisboa.

[3] Berry, M. Kogan, J. (2010). Text Mining: Applications and Theory. Wiley. John Wiley Sons, Ltd.

[4] Bengfort, B., Bilbro, R. and Ojeda, T. (2018). Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning. O'Reilly. O'Reilly Media, Inc.