NOVA UNIVERSITY LISBON

Relevant Expressions and Keywords Automatic Extraction

Author: **Mattia Piccinato, Matteo Saterini**

Academic Year: 2023-24

Contents

Contents **i**

1 Introduction **1**
1.1 Scope . . . 1
1.2 Metrics . . . 1
1.2.1 SCP Metric . . . 1
1.2.2 Dice Metric . . . 2
1.2.3 MI Metric . . . 2
1.3 Information Retrieval . . . 2
1.3.1 TF-IDF Score . . . 2
1.4 Evaluation of Results . . . 3
1.4.1 Precision Measure . . . 3
1.4.2 Recall Measure . . . 3
1.4.3 F1 Score . . . 3

2 First Work **5**
2.1 Stop Words . . . 5
2.1.1 Elbow Method . . . 5
2.1.2 Results . . . 6
2.1.3 Comments . . . 7
2.2 Relevant Expressions . . . 8
2.2.1 Definition . . . 8
2.3 Preliminary Analysis . . . 8
2.3.1 Further Pre-Processing . . . 10
2.3.2 Implementation . . . 11
2.3.3 Filtering . . . 11
2.3.4 Results . . . 11
2.3.5 Evaluation . . . 12

2.3.6 Comments 12

3 Second Work **14**

3.1 Explicit Keywords 14

3.1.1 Definition 14

3.1.2 Glue Choice 14

3.1.3 Implementation 14

3.1.4 Results 15

3.1.5 Evaluation 15

3.1.6 Comments 16

3.2 Implicit Keywords 17

3.2.1 Definition 17

3.2.2 Implementation 17

3.2.3 Optimization 18

3.2.4 Results 18

3.2.5 Comments 18

4 Future work **20**

4.1 First part 20

4.2 Second part 20

5 Bibliography **22**

5.1 References 22

6 Appendix **23**

6.1 Python Functions 23

1 | Introduction

1.1. Scope

This report aims to explore and implement methods for the automatic extraction of relevant expressions and document keywords from textual Big Data corpora, focusing on both explicit and implicit keywords.

These methods require the application of cohesion metrics — essential for evaluating the cohesion of n-grams — as well as measures to quantify the information embodied in n-grams. To ensure better readability, these tools will be introduced in the following sections.

By the end of this study, a comprehensive overview of the techniques used for automatic extraction will be provided, along with an analysis of their performance and effectiveness.

1.2. Metrics

To measure the cohesion within vectors of n subsequent words, we employed three distinct metrics. These metrics are recognized in the scientific literature for their stability across different n-grams and help identify the most meaningful semantic expressions regardless of their length.

1.2.1. SCP Metric

The Symmetrical Conditional Probability (SCP) metric spans values in $[0, 1]$, suggesting higher cohesion for higher values. For any n-gram $w_1, w_2, \ldots, w_n$, it is defined as:

$$SCP(w_1, w_2, ..., w_n) = \frac{p(w_1, w_2, ..., w_n)^2}{\frac{1}{n-1}\sum_{i=1}^{n-1} p(w_1, ..., w_i) \times p(w_{i+1}, ..., w_n)}$$

where $p(x)$ is the probability that x occurs at a given position in any document.

1.2.2. Dice Metric

The Dice metric works similarly to SCP, but it is known for usually (though not always) being less sensitive to a change in frequency for any sub-n-gram. It is defined as:

$$DICE(w_1, w_2, ..., w_n) = \frac{2 \times f(w_1, w_2, ..., w_n)}{\frac{1}{n-1}\sum_{i=1}^{n-1} f(w_1, ..., w_i) + f(w_{i+1}, ..., w_n)}$$

where $f(x)$ is the frequency of x in the corpus.

1.2.3. MI Metric

The Mutual Information (MI) metric for an n-gram ranges from $-\infty$ to $+\infty$, once again suggesting higher cohesion for higher values. It is defined as:

$$MI_p(w_1, ..., w_n) = \log \frac{p(w_1, ..., w_2)}{\frac{1}{n-1}\sum_{i=1}^{n-1} p(w_1, ..., w_i) \times p(w_{i+1}, ..., w_n)}$$

For practical purposes, in our work, we will compute its score using frequencies:

$$MI_f(w_1, ..., w_n) \approx \log \frac{C \times f(w_1, ..., w_2)}{\frac{1}{n-1}\sum_{i=1}^{n-1} f(w_1, ..., w_i) \times f(w_{i+1}, ..., w_n)}$$

where C is the number of words in the corpus.

1.3. Information Retrieval

1.3.1. TF-IDF Score

TF-IDF stands for Term Frequency-Inverse Document Frequency, a statistical measure useful for evaluating the importance of a word in a document within the corpus.

For multi-grams, the formula is composed as follows:

$$Tf - Idf(RE, d_j) = \frac{freq(RE, d_j)}{size(d_j)} \times \log\left(\frac{||D||}{||\{d \in D \wedge freq(RE, d) > 0\}||}\right)$$

The core idea behind TF-IDF is that it balances a term's frequency within the document with its rarity across the corpus.

Indeed, the Term Frequency (TF) assigns higher scores to the most frequent terms in the document, while the role of the Inverse Document Frequency (IDF) is to assign a null score to terms that appear in every document, as they are not specific to any document.

Moreover, it is not sensitive to the size of the document, as the TF factor is normalized by its length, resulting in being a robust measure for our purpose.

1.4. Evaluation of Results

To assess the quality of our results, we leverage the three most-used measures in the literature, namely Precision, Recall and F1-Score.

1.4.1. Precision Measure

Precision is the ability of the algorithm to produce good results and measures their quality. It can be interpreted as the percentage of good results over the total number of results produced:

$$Precision = \frac{tp}{tp + fp} \times 100\%$$

1.4.2. Recall Measure

Recall measures how many of the expected results were found by the algorithm. It is defined as the percentage of the expected results that are successfully retrieved:

$$Recall = \frac{tp}{tp + fn} \times 100\%$$

1.4.3. F1 Score

In general, having high Precision alone does not guarantee that all important keywords are covered, while high Recall does not avoid huge quantities of irrelevant results. Thus, we introduce the F1-Score:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

This new metric ranges in $[0, 1]$ and considers both Precision and Recall equally.

To be mentioned, the scores obtained through the F1 Score are generally closer to the worst of the two measures. For example, if a keyword extraction algorithm identifies 80% of all relevant keywords (Recall = 0.80) but only 50% of its identified keywords are relevant (Precision = 0.50), the F1 score would be:

$$F1 = 2 \times \frac{0.50 \times 0.80}{0.50 + 0.80} = 2 \times \frac{0.40}{1.30} \approx 0.615$$

This way, higher scores of F1 Score are usually associated with higher scores in both Precision and Recall.

In this report, we will mainly consider Precision and F1 Score, as the precision is very important to support our conclusions, and F1 Score is a general value which considers the overall accuracy of our implementation.

2 | First Work

2.1. Stop Words

In this context, stop words are defined as the words $w_1, ..., w_b$ which occur with the most neighbors $Neigh(w)$ throughout the corpus. Essentially, they are words that are not specific to any particular context, such as *of*, *and*, and *in* in the English language.

Stop words are often removed from texts during the pre-processing stages of text mining tasks because they do not carry significant meaning and can introduce noise into the analysis. Filtering them out can improve the performance of various algorithms.

In our work, we extract stop words to design a filter and achieve higher quality results, as explained later in this report.

2.1.1. Elbow Method

The Elbow Method, shown in Figure 2.1, is a heuristic used to identify an appropriate threshold for filtering stop words.

$$b = \text{argmax}_r(|\, f(r + \Delta_k) - f(r) \,| \geq \Delta_k) \qquad \text{where}$$

$$f(r) = NeigSyl(word(r))$$

Δ_k Is a fixed integer distance in the XX axis; 4 proved to be a good value

Figure 2.1: Elbow method

As the distribution of stop words' number of distinct neighbours $Neigh(x)$ follows the curve in the Figure 2.2, the procedure involves identifying a point where the rate of frequency decrease sharply slows down, forming an "elbow." Since the discrete derivative is always negative and keeps increasing while tending to zero, we aim to locate the point where the following words will soon share the number of neighbors with many others, which is where the discrete derivative is approximately 1.

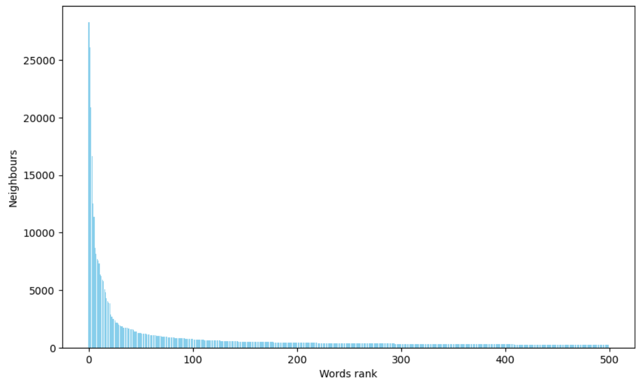

Figure 2.2: Distribution of neighbours for the first 500 words

Based on our analysis, we identified that the most effective value for Δ is 4, as suggested in the papers. This means we should search for the first i-th word w_i such that the difference between $Neigh(w_i)$ and $Neigh(w_{i+4})$ is less than 4. More comments about our analysis and consequent choice will follow our results.

2.1.2. Results

By setting Δ to 4, we identified 242 words as stop words using the dataset *corpus2mw*. This is a reasonable number, considering that most stop word lists for the English language contain between 100 and 600 stop words, depending on the application.

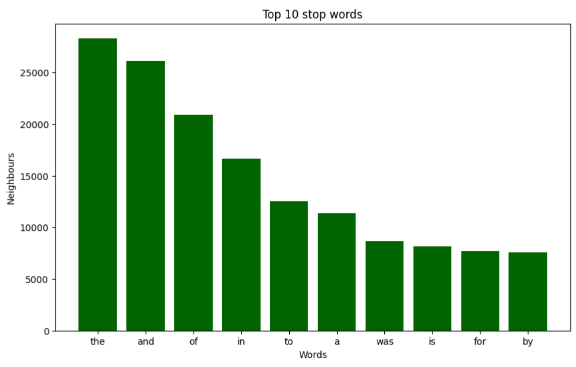

Figure 2.3: Number of distinct neighbours for top 10 stop words

2.1.3. Comments

To determine the best value for Δ, we began by observing that $\Delta = 4$ is likely to be a better value than $\Delta = 1$ as it reduces noise by considering the trend of the next words, rather than just the next one. In addition, we concluded that $\Delta = 4$ is also probably better than $\Delta = 50$ as it provides better discretization, especially where the derivative rapidly changes (around the elbow). Finally, we conducted a search over various values of Δ and found that 4 provides the best trade-off.

We also noted that choosing larger values for the discrete derivative d means being stricter with the definition of a stop word, while choosing a smaller value means going inside the flat zone. Thus, we explored various scenarios, varying the value of d (the value for the discrete derivative) and Δ to find the best value of Δ in each case, as shown in Table 2.1 and Figure 2.4.

		Δ					
		1	2	3	4	5	6
d	0.3333	26	76	83	94	94	119
	0.5000	40	84	94	94	121	121
	1.0000	61	84	123	154	154	154
	2.0000	61	224	176	250	176	250
	3.0000	61	224	284	250	284	284

Table 2.1: Stop words found for each combination

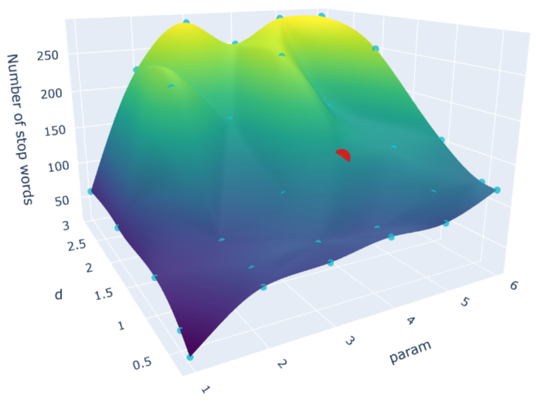

Figure 2.4: Surface of found stop words for each combination

At the end of the analysis, we concluded that using $\Delta = 4$ and $d = 1$ still provides the best trade-off, thus we finally adhered to the standard approach suggested by the papers.

2.2. Relevant Expressions

2.2.1. Definition

In this section, we present the application of the LocalMax technique to extract relevant expressions composed of Multi-Word Units (MWUs) from large-scale corpora.

The idea of the algorithm is to determine which n-grams $w_1, w_2, \ldots, w_n$ can be considered Relevant Expressions. These are defined as expressions that occur at least twice in the corpus and satisfy the following condition:

$$\text{glue}(w_1, w_2, \ldots, w_n) \geq \left(\frac{(\max \Omega_{n-1}(w_1, w_2, \ldots, w_n))^p + (\max \Omega_{n+1}(w_1, w_2, \ldots, w_n))^p}{2} \right)^{\frac{1}{p}}$$

where $\Omega_{n-1}(w_1, w_2, \ldots, w_n)$ is the set of all glues of all the sub (n-1)-grams contained in $(w_1, w_2, \ldots, w_n)$, $\Omega_{n+1}(w_1, w_2, \ldots, w_n)$ is the set of all glues of all the super (n+1)-grams that contain the n-gram $(w_1, w_2, \ldots, w_n)$, and p is some integer value such that $p \geq 1$ (usually $p = 2$).

Essentially, the glue value must be greater than some average of the maximum glue values among its sub-grams and super-grams, representing an "informatively dense expression."

For our work, we assume a stricter definition. According to the adopted definition, the glue must be stronger than for every sub-gram and super-gram. Additionally, each Relevant Expression must occur in two different documents instead of just having frequency ≥ 2, and w_1 and w_n must not be stop words:

$$\text{glue}(w_1, w_2, \ldots, w_n) > \max(\Omega_{n-1}(w_1, w_2, \ldots, w_n) \cup \Omega_{n+1}(w_1, w_2, \ldots, w_n))$$

2.3. Preliminary Analysis

Our first step was to verify Zipf's law and perform the fit for Mandelbrot, by determining the optimal values for both parameters through OLS (Ordinary Least Squares).

As expected, the distribution of words in the corpus adheres to Zipf's law, with $\alpha = 0.815$ (close to 1).

Moreover, it is also evident, just looking at Figure 2.6, that the Mandelbrot model provides an excellent fit for the distribution, with parameters $\alpha = 1.95$ and $\beta = 6.30$.

The distribution of the words with the fit curves and our results about the goodness of fit for the two models, Zipf and Mandelbrot, are shown in the following plots:

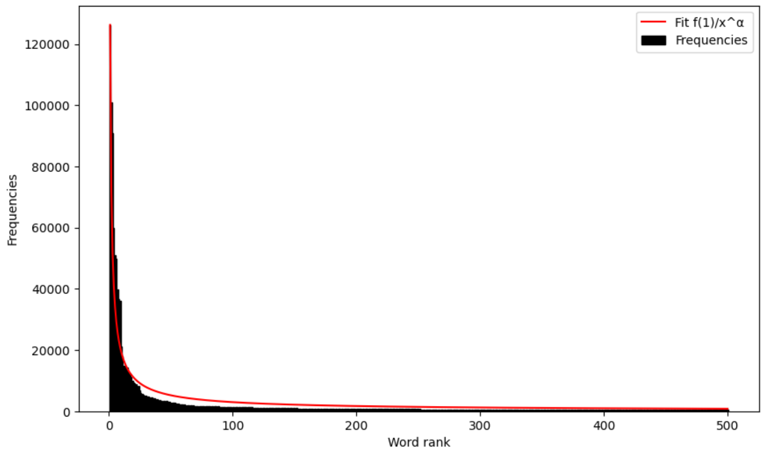

Figure 2.5: Zipf's law

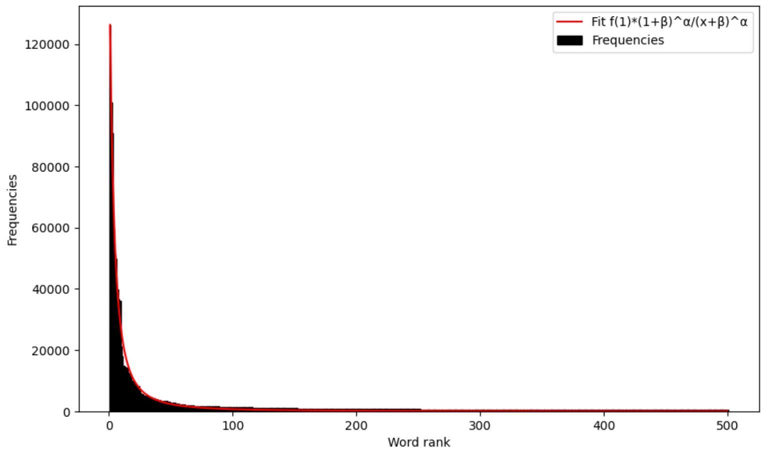

Figure 2.6: Mandelbrot model

Even though the graphical method appears to be sufficient, here we provide statistical evidence as well, to prove that the Mandelbrot model effectively fits the distribution of data excellently:

- Our results for Zipf's model: $f(x) = \frac{f(1)}{x^{\alpha}}$:

$$\text{Root Mean Squared Error (RMSE)} = 182.23$$
$$\text{R squared } (\mathrm{R}^2) = 0.902$$

- Our results for Mandelbrot's model: $f(x) = \frac{f(1)\cdot(1+\beta)^{\alpha}}{(x+\beta)^{\alpha}}$:

$$\text{Root Mean Squared Error (RMSE)} = 61.23$$
$$\text{R squared } (\mathrm{R}^2) = 0.989$$

2.3.1. Further Pre-Processing

Lowercase Corpus

We first converted the entire corpus to lowercase to eliminate case sensitivity, ensuring more accurate identification of word frequencies and patterns, and also reducing the number of distinct expressions, thus improving algorithm efficiency.

Tokenization

Next, we added spaces before and after prominent punctuation marks in each string of the corpus. This separation of words from punctuation reduces noise and ensures reliable word frequency counts. The only exception is the character ', due to the presence of contractions (e.g., *don't* to *do not*). In the following image (2.7) we plot the number of resulting n-grams for each value of n:

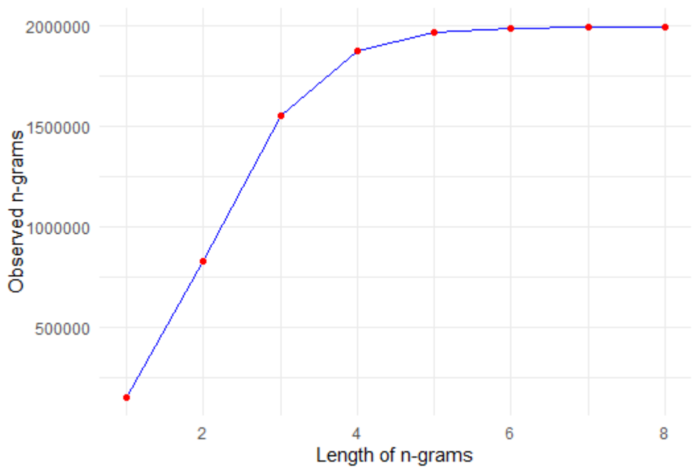

Figure 2.7: Number of n-grams for each value of n

2.3.2. Implementation

The steps of our LocalMax implementation are:

1. Extracting all the single words (tokens) and all the n-grams (with length up to $n = 7$).
2. Storing all the expressions in *n+1* Python dictionaries (one for each length up to $n = 8$), along with their absolute and relative frequencies.
3. Evaluating the glue for each n-gram up to 7-grams, adopting SCP, Dice, and MI metrics.
4. Computing the Ω_{n-1} and Ω_{n+1} of each n-gram).
5. Extracting from the dictionary only the Relevant Expressions according to our definition.

2.3.3. Filtering

As we partially anticipated when introducing the definition of Relevant Expression for our work, we expect that REs respect these two criteria:

- No REs which appear in only one document;
- No REs with w_1 and w_n in the stopwords list.

Moreover, we applied a filter that finally removes the REs with tokens that are just special characters, namely:

- No REs which contain punctation characters **as words**, such as *, . ; : ! ?*.

2.3.4. Results

Table 2.2 presents a sample of Relevant Expressions (REs) extracted from the document *fil_ 15*.

File	REs
SCP	united states, new york, philip ii, south wales, trillion rubles, west virginia, persian gulf, english translation, arab countries, october 2013, lay in repose
Dice	united states, new york, philip ii, south wales, information about, trillion rubles, than any, west virginia, december 1990, york times, persian gulf, english translation, arab countries, october 2013, count of flanders, during this time, lay in repose
MI	persian gulf, english translation, october 2013, new york times,

Table 2.2: Relevant Expressions for fil_15

We observed that, in general, the Dice metric extracts all the expected REs, but it also includes many irrelevant expressions.

Conversely, the MI metric appears to only return relevant expressions, but not all of those we would expect. For instance, in Table 2.2, MI did not extract "west virginia," which we anticipated would be identified. Further analysis revealed that, relative to the corpus *corpus2mw*, $MI(\text{"}west\ virginia\text{"}) = 6.425$, while $MI(\text{"}west\ virginia\ route\text{"}) = 7.385$.

Finally, SCP seems to offer a balance between the other two metrics.

2.3.5. Evaluation

In this section, we will methodically evaluate the quality of the preceding results.

For testing and evaluation, we considered as good REs the ones that satisfy at least one of the following criteria:

- Frozen forms.
- Named entities.
- Very specific *name* - *adjective* forms.

In order to measure the accuracy achieved by our work, we built a test set of 33 REs over a few documents, namely *fil_1114*, *fil_1172*, *fil_1248* and *fil_1340*, and then quantified the quality of our implementation as illustrated in Table 2.3:

GLUE	PRECISION	RECALL	F1-SCORE
SCP	77.42%	72.73%	75.00%
DICE	54.55%	72.73%	62.34%
MI	83.33%	45.45%	58.82%

Table 2.3: REs Extraction Evaluation

2.3.6. Comments

At the end of analysis, comparing our results with the test set, we concluded that, even though our implementation achieved decent results, it is particularly good at finding Named Entities but not as good at finding other forms of REs.

Overall, according to our evaluation, we soon concluded that it would be reasonable to merely consider using either SCP or MI for the second part of our work, as Dice appears to be dominated by SCP, while MI has an average Recall, but has very high Precision.

Even though we will extract the keywords with both SCP and MI, we highlight that MI identifies fewer REs, thus implying faster computation of Implicit Keywords which can often be unfeasible. Moreover, although MI will miss many relevant expressions, it is often enough to find a couple of meaningful keywords to infer the topic of the document. Thus, as we will argue further in the next chapter, MI will be our final choice for the extraction of keywords.

3 | Second Work

3.1. Explicit Keywords

3.1.1. Definition

Keywords are single words or multi-word phrases that describe the main topics of a document, representing the most informative concepts within documents. Explicit Keywords, in particular, occur explicitly in the document.

In this section, we implement an automatic extractor of Explicit Keywords based on the approach explored in the previous chapter.

3.1.2. Glue Choice

As anticipated in the previous chapter, we will use both SCP and MI.

However, we will conclude by arguing that the best choice for *corpus2mw* documents is MI. This statement is just a preview to improve readability, as we will provide reasoning later on.

3.1.3. Implementation

The steps for the extraction of Explicit Keywords are:

1. Compute the Tf-Idf score for each document, for every RE.
2. Compute the Tf-Idf score for each document, for every unigram in our dictionary.
3. Extract up to 10 REs and up to 5 unigrams for each document, ranked by highest Tf-Idf.

This way, every document will have up to 15 keywords, with up to 5 being unigrams and up to 10 being Relevant Expressions. The unigram keywords may be contained within any of the RE keywords.

3.1.4. Results

In Table 3.1, we present a sample of Explicit Keywords extracted from the document *fil_ 3671*.

GLUE	MAIN EXPLICIT KEYWORDS
SCP	metamaterials, constitutive, transformation, coordinate, parameters, constitutive parameters, electromagnetic radiation, san martin
MI	metamaterials, constitutive, transformation, coordinate, parameters, constitutive parameters, electromagnetic radiation, san martin

Table 3.1: Most Relevant Explicit Keywords

As shown, both methods extracted the same keywords for the current document. Indeed, as we will illustrate later, using SCP and MI often yields very similar results, usually with high precision.

However, the keywords *permittivity* and *permeability* are missing, suggesting that the recall of the implementation is not perfect.

3.1.5. Evaluation

In this section, we will methodically evaluate the quality of the preceding results.

For testing and evaluation, we considered any specific word or phrase that summarizes or represents the main concepts or topics covered within the text as an Explicit Keyword in a document.

Here are some examples of keywords:

- In a research paper about climate change: "global warming," "carbon emissions," "climate models," "greenhouse gases."
- In an article about artificial intelligence: "machine learning," "neural networks," "deep learning," "natural language processing."

To test the accuracy of our work, we built a test set of 43 keywords across 5 documents, namely *fil_ 2770*, *fil_ 812*, *fil_ 1173*, *fil_ 3671*, and *fil_ 1635*. The aim was to include different kinds of keywords, such as *content keywords*, *descriptive keywords*, and *subject-specific keywords*.

Subsequently, once the test set was built, we finally quantified the quality of our implementation.

As it is illustrated in Table 3.2, our results show that the low recall for RE extraction using MI as a glue, does not actually affect Explicit Keyword extraction. Indeed, the results obtained when using MI appear to be very similar to those obtained using SCP instead, with MI almost equalizing SCP's Recall and even achieving higher Precision and F1-Score.

Here is the table:

GLUE	PRECISION	RECALL	F1-SCORE
SCP	78.95%	69.77%	74.07%
MI	82.86%	67.44%	74.36%

Table 3.2: Explicit Keywords Evaluation

3.1.6. Comments

Several observations arise about the role of the two glues:

- Both metrics show excellent results in terms of precision and recall, indicating a robust capability of the model in identifying Explicit Keywords.
- It is observed that SCP is no better than MI in terms of F1-score and still provides lower precision. Once again, the recall for SCP is better, but it is often unnecessary to require capturing all the Explicit Keywords of a document.

Thus confirming that MI can achieve results as good as SCP, even though our choice still has to be proven later in this report.

Moreover, regarding the effectiveness of our implementation:

- An important consideration regards the management of base words and derivatives. For instance, the algorithm recognizes "bone" and "bones" as distinct words, even though they refer to the same entity.
- German words such as "Verkehrsbetriebe" and "Verkehrsverbund," meaning "transport service" and "transport network" respectively, are considered as two distinct concepts, even though they share the same root.

These insights suggest possible future work, which will be proposed by the end of the document.

3.2. Implicit Keywords

3.2.1. Definition

In the final step, we aim to identify Implicit Keywords for each document in the corpus. Implicit Keywords are concepts that strongly relate to the content of the document but do not explicitly appear as Explicit Keywords do.

Since concepts are correlated if they frequently appear together in the same documents within a collection, we will extract Implicit Keywords by identifying concepts that are highly correlated with the Explicit Keywords.

3.2.2. Implementation

Our implementation is straightforward and involves searching for Relevant Expressions (REs) in the corpus that do not occur in the current document but are highly correlated with its most significant Explicit Keywords.

Given that the correlation between two REs is defined as follows:

$$Corr(A,B) = \frac{Cov(A,B)}{\sqrt{Cov(A,A)} \times \sqrt{Cov(B,B)}}$$

where the covariance between the two REs is:

$$Cov(A,B) = \frac{1}{||D||-1} \times \sum_{d_i \in D} (p(A,d_i) - p(A,.)) \times (p(B,d_i) - p(B,.))$$

in which $p(A,d_i) = \frac{freq(A,d_i)}{words(d_i)}$ represents the probability of A occurring in d_i, while $p(A,.) = \langle p(A,d_i) \rangle$ is the average over all the documents.

We select the 5 REs with the highest score $\text{Score}(W,d)$ relative to a certain document d:

$$\text{Score}(W,d) = \sum_{i=1}^{n} \frac{\text{Corr}(W,k_i)}{i}$$

According to this definition, our implementation assigns higher score to those REs which highly correlate to the Explicit Keywords with the highest Tf-Idf for that document d, that is, the most relevant for it.

3.2.3. Optimization

The computation of the implicit keywords is usually a very heavy process, due to its quadratic time complexity $O(C^2)$ and to the huge input in the case of Big Data corpora.

To optimize our implementation, we do not actually store the correlation among all the REs in the corpus and the Explicit Keywords of every document in the calculation: instead, we only store it for the pairs of REs such that $\mathrm{Corr}(A, B) > \epsilon$, where ϵ is a small number such as 1×10^{-10}.

This cuts off a lot of computation, significantly reducing the overall required execution time.

3.2.4. Results

In Table 3.3, we present a sample of Implicit Keywords extracted from the document *fil_103*.

GLUE	Implicit Keywords
SCP	dionne warwick, practical application, fell to earth, singles charts, adult contemporary chart
MI	dionne warwick, practical application, fell to earth, sanctuary and chapels, rational animal

Table 3.3: Most Relevant Implicit Keywords for fil_103

As can be seen, both methods extracted the same top three Implicit Keywords, while the fourth and fifth ones differ slightly, being less influential.

This observation appears to be a general trend in our implementation. Indeed, after analyzing 10 different files (considering 50 resulting keywords for each method), namely *1*, *10*, *100*, *103*, *104*, *1007*, *1012*, *1014*, *1106*, and *1151*, the percentage of shared Implicit Keywords using SCP and MI is 54%, while the remaining 46% are not in common.

3.2.5. Comments

Choice of the glue

In the case of Implicit Keywords, both SCP and MI methods consistently find at least 2 or 3 interesting options for each document.

At times, the results provided using SCP are - in our opinion - more interesting than those provided by using MI. The reason is likely due to the fact that using MI as a glue yields a smaller number of REs extracted from the corpus, resulting in fewer words to look for high correlation with. However, this is also why the computation is faster using MI rather than SCP.

Therefore, since MI will always provide 2 or 3 interesting Implicit Keywords, we believe that adopting it instead of SCP will allow the user to explore even larger corpora than *corpus2mw*, which may take longer using SCP, without sacrificing meaningful results.

Effectiveness of the algorithm

Based on our analysis, we believe that the significance of the top Implicit Keyword "Dionne Warwick" for file *fil_103* (see Table 3.3) deserves closer inspection, in order to show the effectiveness of the algorithm.

Dionne Warwick is an African American singer, notably appearing as a guest on "The Tonight Show" hosted by Harry Belafonte, a prominent African American journalist. The document in question mentions Belafonte's involvement in various musical projects that, as described in the document, were associated with significant events such as the Grammy Awards. Hence, the inclusion of *Dionne Warwick* as an Implicit Keyword is particularly fitting, given the context of discussions around Belafonte, Black Music, and the Grammy Awards, even though the correlation between *Dionne Warwick* and *Harry Belafonte* only amounts to 0.515. This shows that the shared Implicit Keywords and the specific relevance of *Dionne Warwick* underscore the effectiveness of the algorithm in capturing key themes and concepts within the document.

4 | Future work

4.1. First part

Our implementation showed good results in correctly retrieving a decent quantity of Relevant Expressions from a Big Data corpus.

Here are some further improvements regarding the first part of the work which could be carried out to achieve even better results:

- Since the LocalMax algorithm does not take into account the length of Multi-Word Units for them to be considered Relevant Expressions, ending up in prioritizing short Relevant Expressions, it could be reasonable to indeed take into consideration the number of words in the RE. Specifically, in future applications, we propose incorporating prior probabilities based on the length of the phrases. We believe that employing a Bayesian approach could significantly enhance the precision of 6-grams and 7-grams.
- The syllables count SylCo could also serve to improve the performance of our implementation. Indeed, introducing the concept of NeigSyl, defined as the ratio between a word's BigramNeig (neighboring two grams) and its SylCo, will probably lead to a more accurate identification of stop words, typically characterized by a lower syllable count. Regarding this matter, we invite the reader to check out our proposal for *SylCo*) in the appendix, at the end of the document.

4.2. Second part

Regarding the second part of the work, here are our indications:

- In order to improve the quality of the Implicit Keywords, aiming at achieving at least 5 good quality Implicit Keywords for each document, we suggest to take Intra-Document

Proximity into consideration, instead of only considering Inter-Document features.

- In order to improve the precision of our algorithms, the next work should take into account the similarity between base words and their derivatives, as we believe that it will help identifying the Explicit and Implicit Keywords of each document.
- As an additional feature, it could be worth the effort to exploit our implementation to cluster and/or classify the corpora documents.

5 | Bibliography

5.1. References

Papers

- Stopwords in Technical Language Processing - Serhad Sarica and Jianxi Luo
- A Theoretical Model for n-gram Distribution in Big Data Corpora - Joaquim F. Silva, Carlos Goncalves and Jose C. Cunha
- Document Clustering and Cluster Topic Extraction in Multilingual Corpora - Joaquim Silva and Joao Mexis
- Improving LocalMax Multiword Expression Statistical Extractor - Joaquim F. Silva and G. P. Lopes
- Stackoverflow post - Sylco Function

6 | Appendix

6.1. Python Functions

token

The `token` function splits a word into tokens, considering special characters separately. It handles different cases based on the length of the word and the presence of special characters at the beginning or end of the word.

create_list_of_dict

The `create_list_of_dict` function constructs a list of dictionaries, each representing n-grams of various lengths (from 1 to a specified maximum length). Each dictionary entry maps an n-gram to a list that includes the absolute frequency of the n-gram, the indices of the documents where it appears, and the relative frequency within those documents. This function is essential for efficiently storing and retrieving n-grams for text analysis tasks.

get_stop_words

The `get_stop_words` function extracts stop words from a corpus by analyzing the frequency of neighboring words. It identifies words that appear frequently at the boundaries of documents or between words, and filters out special characters. The function then determines the elbow point to separate common words from less frequent ones, effectively identifying stop words for the corpus.

compute_glues

The `compute_glues` function calculates the cohesion of n-grams using different glue metrics, such as Dice, Symmetrical Conditional Probability (SCP), and Mutual Information (MI). It iterates over n-grams of various lengths and computes the specified glue value for each n-gram based on its frequency and the frequencies of its sub-components. The function returns a list of dictionaries, each containing the glue values for the corresponding n-grams.

find_RE

The `find_RE` function identifies relevant expressions (REs) from a set of glue values calculated for n-grams. It processes each n-gram to determine if it meets the criteria for being an RE, and if so, includes it in the resulting dictionaries. This function is crucial for filtering out less significant n-grams, focusing on those with higher relevance for further analysis.

create_tfidf_and_probs

The `create_tfidf_and_probs` function generates two lists of dictionaries, each containing the Term Frequency-Inverse Document Frequency (TFIDF) values and probabilities for n-grams. This function initializes two empty lists to store these dictionaries and iterates over each n-gram to compute the TFIDF and probability values based on their frequencies and occurrences in the corpus. The resulting lists of dictionaries provide a detailed representation of the statistical significance and distribution of each n-gram within the corpus, which is crucial for tasks like text mining and natural language processing.

find_explicit_keywords

The `find_explicit_keywords` function identifies and returns explicit keywords for each document in the corpus. It also provides relevant expressions (REs) that contain at least two words. The function initializes dictionaries for unigrams and multi-word expressions, populates these dictionaries with corresponding TFIDF values, and sorts them based on their significance. By combining unigram and multi-word explicit keywords, the function delivers a comprehensive set of key terms for each document, aiding in the extraction of meaningful insights from the text.

create_corr_dict

The `create_corr_dict` function computes the correlation dictionary for relevant expressions (REs) based on their probability distributions across documents. It utilizes a helper function, `cov`, to calculate the covariance between two REs. The function iterates through all pairs of REs to determine their covariance scores, then normalizes these scores to compute the correlation values. The resulting dictionary provides a measure of how strongly each pair of REs is related, which is useful for understanding the semantic relationships between terms in the corpus.

find_implicit_keywords

sylco

The function `sylco` is used to calculate the number of syllables in a given word. The function utilizes various rules and exceptions to determine the syllable count, including considerations for vowels, endings like "es" and "ed", consecutive vowels, prefixes like "co-" and "pre-", and exceptional words. The function employs regular expressions for pattern matching and string manipulation techniques. Additionally, it returns a syllable count, ensuring a minimum of 1 syllable for words with less than 3 letters. The code demonstrates a systematic approach to syllable counting and provides flexibility to handle various word forms and exceptions.