POLITECNICO DI TORINO

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Keyword Extraction and Classification

Relatore
Prof.ssa Silvia Anna Chiusano
Co-Relatore
Prof. Raphaël Troncy
Tutor Aziendale
Viktor Botev

Candidato
Eugenia Spano

Anno Accademico 2017-18

SUPERVISORS:
Botev Viktor, IRIS.AI, Göteborg
Chiusano Silvia Anna, Politecnico di Torino, Torino
Troncy Raphaël, Eurecom, Sophia Antipolis

LOCATION:
Göteborg

ABSTRACT

The first part of this thesis focuses on Keyword Extraction, with the goal of selecting the words that represent the "true" meaning of the text. Inspired by the PositionRank algorithm, in order to overcome its limitations in some specific cases, a TF-IDF-biased version of the TextRank algorithm is implemented and used to extract keywords. The results show that the system here proposed constitutes a valid alternative to the PositionRank algorithm, and in some cases it even outperforms the TextRank baseline. Going further, in the second part of the thesis, an LSTM Neural Network is used to classify the extracted keywords as belonging to either the problem, solution, evaluation or results section. Not only will this provide some contextual information about the keywords, but it will also highlight the structure of each paper, allowing for new possibilities of analysis.

ACKNOWLEDGEMENTS

The biggest thank you goes to my family. To *Mamma e Babbo*, for your constant love and support. I would certainly not have made it here if it wasn't for you. To *Lor*, for trusting me enough to go to "Cala Goloritzè", for finishing my "100 montaditos", for making me laugh like nobody else does.

Thank you *Oskar*, for staying by my side every step of the way, and always believing in me.

I would also like to express my sincere gratitude to my supervisor, *Viktor*. Thank you for sharing your knowledge with me and guiding me through this journey. It has been a pleasure working with you.

CONTENTS

1 INTRODUCTION 1
1.1 Background . . . 1
1.2 Goal and Contribution . . . 2
1.3 Thesis outline . . . 3
2 THEORETICAL OVERVIEW AND RELATED WORK 5
2.1 Keyword Extraction . . . 5
2.1.1 Pre-processing and Candidate Selection . . . 5
2.1.2 Ranking/Classification . . . 6
2.1.3 Extraction . . . 7
2.1.4 Post-processing . . . 7
2.2 Keyword Classification . . . 7
2.3 Related Work . . . 8
2.3.1 Frequency-based solution . . . 8
2.3.2 Machine-learning-based solutions . . . 9
2.3.3 Graph-based solutions . . . 10
2.3.4 Alternative solutions . . . 11
2.3.5 Classification . . . 12
3 THE APPROACH 13
3.1 Mathematical Framework . . . 13
3.1.1 The PageRank algorithm . . . 13
3.1.2 The TextRank algorithm . . . 14
3.1.3 The TF-IDF bias . . . 15
3.1.4 Modifications to the original algorithms . . . 15
3.2 Algorithmic Framework . . . 16
4 METHODOLOGY 19
4.1 Datasets . . . 19
4.2 Evaluation Metrics . . . 19
4.3 The algorithms . . . 20
4.3.1 Baseline algorithms . . . 21
4.3.2 Biasing algorithms . . . 23
4.3.3 Neural Network . . . 25
4.4 Experimental Setup . . . 25
4.5 Experiments . . . 26

4.5.1 Starting hypothesis . . . 26
4.5.2 Baseline choice . . . 27
4.5.3 Bias analysis . . . 27
4.5.4 Neural Network parameter tuning . . . 28
4.5.5 Training the Neural Network . . . 28
4.5.6 Experiment on Keyword Classification . . . 30
5 RESULTS 31
6 COMMENTS AND DISCUSSION 37
6.1 Biased PageRank: the reason behind it . . . 37
6.2 Analysis of the results for Keyword Extraction . . . 37
6.3 Analysis of the results for the Keyword Classification . . . 41
6.4 Threats to validity . . . 41
7 FINAL REMARKS AND FUTURE WORK 43
7.1 Conclusions . . . 43
7.2 Future Work . . . 43
A APPENDIX A: DATASETS 45
A.1 Documents in *Small_full_texts* dataset . . . 45
A.2 *IRIS_full_texts* dataset format . . . 45
A.3 Documents for LSTM evaluation . . . 46
B APPENDIX B: RESULTS 49
B.1 Evaluation of results: beyond the numbers . . . 49
B.2 Results for the classification task . . . 49
BIBLIOGRAPHY 53

LIST OF FIGURES

Figure 2.1 Keyword Extraction Process 6
Figure 3.1 Main steps of the complete algorithm 18
Figure 4.1 Creation of LSTM input 30
Figure 5.1 Comparison of baseline algorithms on *IRIS_abstracts* dataset. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2. 31
Figure 5.2 Comparison of the biased TextRank with the baselines on abstracts. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2. 32
Figure 5.3 Comparison of the biased TextRank with the baselines on full texts. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2. 33
Figure 5.4 Performance of the PositionRank algorithm in case of full texts, full texts with no abstract, and in case the ground truth keywords were chosen to be less generic. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2. 34
Figure 5.5 Comparison of the biased TextRank with the baselines on full text after the removal of the abstract. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2. . 35
Figure 5.6 Comparison of the biased TextRank with the baselines on full texts, modifying the ground truth keywords to be less general. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2. 36

LIST OF TABLES

Table 4.1 Datasets . . . 19
Table 4.2 List and usage of the main Python libraries . . . 26
Table 4.3 Neural Network parameter names and tested values. In bold the chosen values. . . . 29
Table 4.4 Section Types and Characterizing Words . . . 29
Table 6.1 Example of modified list of keywords on *Small_full_texts* dataset . . . 39
Table 6.2 Difference in F-measure when relaxing constraints on the comparison of keywords. Matching keywords indicated in bold. . . . 40
Table B.1 Example of Keyword Extraction performed with the TF-IDF biased TextRank on a sample text taken from *IRIS_abstracts* dataset. F-measure: 30%. In bold the extracted keywords, at the bottom the true keywords. The correctly identified keywords are underlined. . . . 49
Table B.2 Example of Keyword Extraction performed with the TF-IDF biased TextRank on a sample text taken from *IRIS_abstracts* dataset. F-measure: 0% In bold the extracted keywords, at the bottom the true keywords. . . . 50
Table B.3 Detailed results for the classification task (1/2). P: Problem, S: Solution, E: Evaluation, R: Results. . . . 51
Table B.4 Detailed results for the classification task (2/2). P: Problem, S: Solution, E: Evaluation, R: Results. . . . 52

ACRONYMS

IDF Inverse Document Frequency

LSTM Long Short Term Memory

NN Neural Network

POS Part Of Speech

RAKE Rapid Automatic Keyword Extraction

RNN Recurrent Neural Network

SVM Support Vector Machine

TF Term Frequency

1 INTRODUCTION

1.1 BACKGROUND

Given the significant amount of scientific papers that get published every day, it has become increasingly difficult to stay up-to-date with the latest research. It is necessary to organize this huge quantity of information, in a way that allows easy identification and retrieval of only the documents of interest. Various approaches have been proposed to tackle this problem. Automatic indexing and automatic filtering, for example, allow the retrieval of all the documents that match with specific query terms provided by the user [1, 2]. Automatic summarization allows for reduction of the amount of words in a document yet preserving all its information. This yields a reduction of the time needed to analyze the content of the document and determine if it is of interest [3]. Classification and clustering, that help in classifying or grouping documents into specific categories, could be another approach to make sense of all that information. During classification, each document is assigned a label chosen from a predefined set, so that similar documents will be easily recognized by having the same label. When clustering texts, the goal is to group them so that documents within the same group will share common characteristics. After having identified one document as relevant, it would then be easy to mark all the other documents with the same label or in the same cluster as relevant as well [4, 5]. Last but not least, topic detection allows for selection of documents based on their topic, which is another criterion to determine whether or not a document is of interest [6].

In general, one could consider the presence of keywords to be beneficial in any task that requires to have an overall understanding on the content of a document. In fact, keywords could be considered to be the "true" meaning of the text, which in this context, has been chosen to be the smallest set of words that preserves all the information contained in the text itself. The process of *Keyword Extraction* aims at extracting words that best represent this "true" meaning, and as such, it represents another way to fasten the retrieval of relevant documents. In fact, it would be enough to analyze the keywords to know what the document is about, and it would be enough to compare keywords to find similar documents, hence providing a quick way to select relevant ones. After extracting keywords, it is also possible to classify them, through the process of *Keyword Classification*. For example, by classifying keywords as belonging to the problem, solution, evaluation or results section, one would be able to determine if two

or more documents have similar problems or reach the same conclusion. In this work, both Keyword Extraction and Classification will be explored. In order to evaluate the automation of these processes, it is necessary to have a gold standard to compare the results to, but in many cases this data does not exist. Despite the fact that identifying keywords in texts is of vital importance for many Text Mining (TM), Information Retrieval (IR) and Natural Language Processing (NLP) tasks [7], the vast majority of the authors still do not include a set of keywords in their work. In addition to this, a much bigger problem is that even when they do include them, there still is no standardized method for evaluating the results, meaning there is no globally-recognized way to verify that this is actually the smallest set of words that still preserves the information in the text. Furthermore, when the same text is given to different people, it is highly probable that the words one considers to be keywords might be different from the ones selected by another person. The high subjectivity of the task makes it particularly difficult to have a ground truth to evaluate and compare new works with. On top of all this, the huge volume of data to deal with makes it impossible to manually carry out the keyword assignment process. It is important to stress though that even on smaller sets of documents it is still difficult for humans to create a ground truth, and this makes it even harder for a machine to perform well, having no ground truth to learn from. That is why the topic of Keyword Extraction and Classification is currently an open issue, and that is also why, as will be shown in Section 2.3, a considerable number of people have been focusing on finding a way to automate this process.

This thesis was proposed by IRIS.AI, a company that developed a system for automatic recommendation of scientific papers. Their A.I. is useful both for individuals and companies who can benefit from a tool that speeds up their research process. The engine is in fact able to understand the content of the papers it's presented with, in order to propose interesting similar works to the user. The work here presented will help them refine and improve their recommendation engine by allowing to add more features to it.

1.2 GOAL AND CONTRIBUTION

The goal of the first part of this thesis is to find a way to extract keywords from scientific texts, in order to provide a good overview of the different documents. Note that, for this work, all the documents are assumed to be in English, unless clearly stated otherwise. The main contribution is a new algorithm that uses word embeddings together with classical keyword extraction algorithms such as TextRank [8] to efficiently extract the "true" meaning of the text. Going further, in the second part of the thesis the extracted keywords will be classified and grouped into 4 categories, which represent the building blocks of every scientific paper: problem, solution, evaluation and results. By doing this, it will be possible to give ad-

ditional information about each paper, on a much finer granularity level than just providing the keywords. The structure of each document will be easily accessible, which will open the doors to numerous new possibilities of analysis. For example, having access to the structure of documents allows their comparison on a section-level: as previously mentioned, it could be useful to compare documents based on the problem they are trying to solve or the solution they propose. It is also interesting to mention that keyword extraction alone would not be sufficient to determine in which context a keyword was used. With the additional classification step, a keyword like "dimensionality reduction", that could have been used both to describe a possible implementation technique and a potential problem, would be instead assigned to one category only, hence providing some contextual information.
The work can be divided into two subtasks: the first subtask will focus on finding a way to (efficiently) extract keywords from scientific articles, while the second subtask will consist in grouping the extracted keywords into the aforementioned 4 categories.

1.3 THESIS OUTLINE

The rest of the thesis will be structured as follows. Chapter 2 will provide some theoretical background on Keyword Extraction and Classification, together with an overview of the related work on the topic. Chapter 3 will present the solution adopted, focusing on both the mathematical and algorithmic point of view. Chapter 4 will describe the evaluation metrics and the experiments conducted, together with information about the dataset used, and the description of the experimental setup. Chapter 5 will present the most relevant results. Finally, Chapter 6 will focus on commenting the work and discussing the results and Chapter 7 will explore possible future work.

2

THEORETICAL OVERVIEW AND RELATED WORK

As mentioned in Section 1.1, extracting keywords is highly beneficial for many applications. In fact, having keywords available helps to have an overview of what the text is about, which in turn helps in selecting only the documents of interest. Moreover, the additional step of classifying the keywords highlights the structure and composition, thus providing further insight on the document. Given its importance, this topic has been extensively studied and researched, hence explaining the vast literature available. Despite its popularity though, some questions still remain unanswered, such as how to evaluate the results or how to provide a ground truth given the level of subjectivity of the task, making it a currently open issue. The importance and vast application of Keyword Extraction and Classification explains this being the central topic of the thesis. In particular, this chapter will focus on explaining how the extraction and classification of keywords is usually performed in the literature.

2.1 KEYWORD EXTRACTION

The Keyword Extraction process aims at obtaining a list of words, ranked in order to show which ones are more likely to be keywords and which ones are not. The ultimate goal is to provide the minimum number of words that best describe the text without significant information loss. More formally, given a document D containing N words ($D = [w_1, w_2, ..., w_N]$), the Keyword Extraction process aims at finding the smallest subset $K \subseteq D$ of words that still preserve the meaning of the text. Inspired by the work of Lahiri et al. [9] who identified 3 main phases in the process of Keyword Extraction, a similar division is presented here: first a *Pre-processing* and *Candidate Selection* phase, in which the potential words to be keywords get selected, followed by a *Ranking/Classification* phase, in which the selected candidates are given a score which reflects the probability of being a keyword. Then an *Extraction* phase is added, despite not being present in [9], to give more details about the criteria used to select the final keywords, and finally a *Post-processing* phase, which constitutes an optional step in this process. All the steps and their inputs and outputs are summarized in Figure 2.1.

2.1.1 *Pre-processing and Candidate Selection*

The *Pre-processing* phase aims at preparing and transforming the text, in order to have it in a format which is convenient for the analysis. In fact, the "raw" text contains many words that carry no real meaning, and should therefore be removed. This also reduces the dimensionality of the problem space, which is always encouraged in order to significantly cut down the computation time and the required storage space. In addition to this, stemming and lemmatization [10] are usually performed on the remaining words, so that all variations of the same word (inflectional forms and derivationally related forms) can be counted as occurrences and not as different terms. The importance of this step can be immediately seen anytime a count-based or frequency-based algorithm is applied to a piece of text, but it is also performed even in presence of graph-based algorithms and Neural Networks (NN). In fact, graph-based methods rank the nodes in the graph similarly to frequency-based methods, while Neural Networks benefit from stemming and lemmatization because it reduces the number of inputs, allowing for a speed-up in the training time.

The next step is the *Candidate Selection* phase. The typical strategy for selecting candidate words consists in using a Part of Speech tagger (POS tagger) to filter out some of the terms. A POS tagger, in fact, is able to assign each word to a part-of-speech category, such as adjective, verb, noun etc., which allows the user to easily spot and select the words to keep as candidates. Typically the candidate keywords are going to be chosen among nouns and adjectives only, as it has been shown that they represent the most frequently occurring POS tags in the manually assigned keywords [11].

Figure 2.1: Keyword Extraction Process

2.1.2 *Ranking/Classification*

The *Ranking/Classification* phase aims at finding a criterion to identify which of the words have a higher chance of being a keyword. This could be achieved either by classifying the words

as keywords or non-keywords, or by scoring the words so that the score would reflect the chance that word is indeed a keyword. The different criteria can be based on the frequency of words in the text, can rely on machine learning algorithms, or can be based on a graph representation of the text. An overview of the literature available can be found in [3, 12].

2.1.3 *Extraction*

After the *Ranking/Classification* phase is completed, the *Extraction* phase is comparatively quite easy, in that it only involves the selection of the top n words based on the ranking previously performed, or, in case of classification, the extraction of the words classified as being keywords. As previously stated, the score given to a word indicates how likely that words is to be a keyword, hence the higher the score, the better the word. A common issue in this case is how to determine the value of n. It usually is derived as a function of the number of words in the text [8], but it could also be chosen experimentally after trying different values [13].

2.1.4 *Post-processing*

The post-processing phase can be performed if necessary. It might consist in the removal of synonyms and/or variations of the same word from the list of selected keywords [14], or it might involve the creation of keyphrases, as a result of the merging of two or more keywords which appear consequently in the text [8, 13].

2.2 KEYWORD CLASSIFICATION

The logical continuation of this work, after extracting the keywords, is to classify them. This section will cover the basics of classification, by first providing a definition and some common algorithms, and then diving into the details of the more pertinent topic of text classification. The term *classification* or *categorization* indicates the process that allows to assign a specific piece of data to a pre-determined category. Formally, given a set of training records $X = x_1, ..., x_N$ such that each is labeled with a class value chosen from a pre-determined set $C = c_1, ..., c_k$, a classification model is constructed, which will be able to classify test instances whose class value is unknown [15]. It is important to mention that the assignment of a test instance to a class can be a *hard assignment*, if the instance is assigned to one and only one class, or a *soft assignment*, if the test instance is assigned to a class with a certain probability. Text

Classification (or Categorization) is the category to which the process of *Keyword Classification* belongs. As the name suggests, the goal of Text Categorization is to classify texts, hence to provide a label that identifies the specific document. Inspired by the definition of Text Categorization found in [4], one could state that Keyword Classification consists in assigning a boolean value to each pair $(w_j, c_i) \in W \times C$, where $D = w_1, ..., w_{|W|}$ represents the list of words in a text and $C = c_1, ..., c_{|C|}$ is the list of pre-defined categories. A value of T (True) will be assigned to the pair (w_j, c_i) whenever the word w_j belongs to the category c_i, a value of F (False) otherwise. Multiple classification algorithms are suitable for the task, such as Decision Trees, Support Vector Machine (SVM) classifiers, Neural Networks and Bayesian classifiers [15]. The focus of this work will be put on Neural Networks, in particular on Long Short Term Memory (LSTM) Networks, which are able to handle sequential inputs and are therefore suitable for texts.

2.3 RELATED WORK

This section aims at presenting what has been done in the literature, for both keyword extraction and classification. Concerning keyword extraction, as mentioned in the previous section, a vital step is to find a way to score and rank the words in the text. Depending on how this step is performed, the different solutions can be divided in 4 categories: frequency-based, machine-learning-based, graph-based and alternative solutions.

2.3.1 *Frequency-based solution*

One of the oldest yet effective methods for extracting keywords was based on the assumption that the more frequent a word is in a document, the higher the probability that that word is important for that document. This is known as TF (*Term Frequency*) and it is a simple occurrence count on each term in the text. Considering only the term frequency though has some limitations, since there are words that are generally more frequent than others, and therefore the fact that they are found to be frequent inside a specific document is not to be considered as something out of the ordinary. That is why people came up with what is commonly referred to as TF-IDF (*Term Frequency-Inverse Document Frequency*).

$$w_d = f_{w,d} \log \left(\frac{|D|}{f_{w,D}} \right)$$

In its simple formula, the words that are generally frequent (those whose occurrence count f_w is high in a given corpus D) get somehow penalized, so that only the words that are

representative for the specific document will obtain a high score w_d (see Section 3.1.3 for more mathematical details). The simplicity of the formula and the fact that it yields competitive results are the reasons why it is still used nowadays and why it is included in this chapter. Despite the fact that it is quite an old formula [16], it wasn't until the 2000s that it was used as an actual way to identify important terms. In 2003 for example, Juan Ramos used TF-IDF to find which words in a document were most suitable to be used as query terms [17]. In 2010, in order to solve the SemEval task 5 "Automatic Keyphrase Extraction from Scientific Articles", Pianta and Tonelli created *KX: A flexible system for Keyphrase eXtraction* [18], which used TF-IDF to score each word and rank them accordingly, yet providing flexibility by letting the user be able to choose some of the thresholds involved in the process. These are two examples of how TF-IDF has been used directly to solve the problem of Keyword Extraction, but there are many other cases in which it has been used indirectly, usually as a feature that contributed to the score of the words, which in turn was part of a more complicated algorithm.

2.3.2 *Machine-learning-based solutions*

This category of algorithms aims at solving the Keyword Extraction problem by exploiting machine learning techniques. An example can be found with KEA [19], that turns the Keyword Extraction task into a classification problem. This is also an example of the use of the TF-IDF score as a feature, instead of a direct method to rank words. More in details, KEA utilizes the Naïve Bayes algorithm to create a model capable of identifying keyphrases in a document. It requires a training set where the keywords and keyphrases are known. For each training document, it identifies candidate phrases and calculates their feature values (TF-IDF and first occurrence position). Each phrase is then marked as keyphrase or non-keyphrase, which represent the classes of the model. After the model is created, in order to extract new keywords from unseen documents, they are input into the model. Then, the process of extracting candidates and calculating features is performed. After that, KEA calculates two probabilities $P[yes]$ and $P[no]$ (respectively the probability of a word being or not being a keyword), and ranks each word by the final score $p = P[yes]/(P[yes] + P[no])$. Some post-processing is then computed on the ranked words to decide which one to select. Even in [20] the TF-IDF was used as one of the features to create a vector representation of each word in the document. In this case though, the authors used the feature vectors as input in a Neural Network model that was able to classify each word as Keyword or Non-Keyword.

2.3.3 *Graph-based solutions*

Another set of approaches include those that rely on a graph representation of the document to be able to infer properties about the text. The idea to use a graph structure to infer information and properties of texts has a long history. Back in 1998, Kleinberg proposed *HITS*, an algorithm capable of ranking web pages as a result of a search query. Right after this, a milestone in the graph history was set by Page et al. with the famous *PageRank* algorithm [21], which presented another effective way to rank web pages. This algorithm was able to overcome the major limitation of HITS, which would only work on a limited subset of pages. PageRank was based on the idea that the entire World Wide Web could be represented as a directed graph, in which every vertex indicated a web page, and every directed edge indicated a link pointing from a page to another. Given this representation, every page was given a score that recursively depended on the scores of the pages that pointed at it. These two algorithms can be easily adapted to the problem of Keyword Extraction: instead of web pages and links, the graph would contain the words and the connections between the words in the text. Considering this simple modification, both algorithms are extensively used in the literature, so much that they could be considered to be the foundation of the graph-based approaches for Keyword Extraction, in that almost every other method later developed is to some extent based on them.

The perfect example is *TextRank* [8]. TextRank can be considered to be the baseline for this category, being the first algorithm to use a graph representation of text as the basis for extracting keywords. Each noun and adjective of the text is added as a vertex of the graph, and an edge is added between two terms if they co-occur within a context window of N words. After the graph is constructed, each vertex is given an initial value of 1, and the PageRank algorithm is applied until a threshold is reached. The top T words are then selected for further processing, where T is chosen depending on the number of vertices in the graph.

Even in more recent years, one can still find that PageRank has an influence on the proposed solutions. In July 2017, a paper was published which presented *PositionRank* [13], again based on the work of Page et al., which was able to achieve state-of-the-art results. The key idea behind this work is that the authors believe that keywords are found at the beginning of a document rather than at the end. Therefore, they propose a variation of the basic PageRank formula, that would allow them to give more importance to the words which were found earlier in the document. This way all the occurrences of each word in the text are taken into account, but the biggest contribution is given from the earlier occurrences rather than the later ones.

Finally, although not based on PageRank, it is worth to mention the *RAKE* algorithm. The particularity of this algorithm resides in the fact that it has been proven to be over six times

faster than TextRank in extracting keywords [22]. It uses stopwords and phrase delimiters to extract candidate keywords from the text. It then construct a co-occurrence graph of all the words in the text, where each vertex represents a word, and each undirected edge is weighted with the number of times that the two words co-occur in the document. Three scores are then calculated for every word: the degree of the word, which considers the number of edges incident to that vertex, the frequency of the word, which counts the occurrences in the text, and the ratio between the two. Candidate keywords that are composed of more than one word get the sum of the score of the single words as final score. The candidate keywords are then sorted choosing one of the three score metrics proposed, and the top T words are selected as keywords, where T is one third of the number of candidates.

2.3.4 *Alternative solutions*

To conclude the overview on all the possible solutions that have been proposed to solve the task, here follow the so-called "alternative approaches". These approaches are valuable because, despite the fact that they haven't always had a huge follow-up in the research world, they allow to look at the problem from a completely different point of view.
For example, Yang et al. in 2013 published *Keyword extraction by entropy difference between the intrinsic and extrinsic mode* [23]. Inspired by the physics domain, their idea was to use the concept of entropy difference between the intrinsic and extrinsic mode to calculate a score that could then be used to discriminate between the different words of the text. The reasoning behind this work is that a common word (non-keyword) is usually found evenly spread inside the document, while this does not apply to keywords, which tend to be concentrated in certain areas of the text and have a non-uniform distribution. This being said, the authors calculated the intrinsic and extrinsic mode entropy and used their difference as the score by which the words were ranked. The higher the score, the higher the probability of the word being a keyword. A different approach consists in exploiting word embeddings. Word embeddings were first introduced by Mikolov et al. in 2013 [24], who presented *word2vec*, a model capable of capturing relations between words, not only their meaning, through their vector representation. In 2015, Kusner et al. presented the Words Mover's Distance (WMD) [25], a function designed to calculate distances between documents. In this work, vector representations are exploited in order to calculate the distance between words, and the document dissimilarity is computed as the distance that words in a document have to "travel" to reach the word vectors of the other document. Inspired by their work, the approach here proposed consists in using the vector space and vector representation of words to find keywords.

2.3.5 *Classification*

For what concerns the Keyword Classification part, it is hard to come by previous works that deal solely with that problem. One the other hand, since as stated at the beginning of this chapter Keyword Classification is a particular case of Text Classification, one could refer to this category of works in order to have an idea of what has been done in that field. More generally, Classification can be applied to different tasks. A typical example is spam detection, after which an email is classified as being spam or not spam [26]. Another area of research on the topic concerns image classification. For example, in [27], it allowed the detection and analysis anomalies found in digital images that could lead to breast cancer. Sentiment analysis, which is another possible application, helps indicating if reviews express positive, neutral or negative opinions [28], and it could be considered as a particular case of text classification, which aims at classifying written documents or texts. A final example of classification can be found in [29]. The model presented aimed at classifying the sentiment of a document as either positive, negative or neutral. To do so, the authors use an LSTM Neural Network, in combination with a Convolutional Neural Network to learn the semantics of the sentences that compose each document.

For the Keyword Extraction task, the thesis will be centered on graph-based methods. In particular, TextRank was chosen as the base algorithm, since it was the first algorithm to propose a graph representation of texts to solve the Keyword Extraction problem and it showed promising results compared to the other methods. Inspired by the state-of-the-art results and method proposed by Florescu et al. in PositionRank, which is build on top of TextRank using a word position bias, several biasing criteria were explored and implemented, in order to understand which of those yielded the best results. Among them, TF-IDF, due to its simplicity and effectiveness was able to produce the best results and is going to be explored in more details in the next chapter. For the Keyword Classification task, a Long Short Term Memory (LSTM) Network was implemented. The reason for this type of NN lies on the fact that it is a particular type of Recurrent Neural Networks (RNN) which is able to capture the relationship between inputs, even when they are distant in time. This works perfectly with texts, since often, in the English language, we find that two words, which are highly correlated, do not appear close to each other. It is necessary for the network to have some kind of *memory* of past inputs, in order for that correlation to be learned.

3

THE APPROACH

The proposed idea will look at the problem as two subsequent parts of a process: first the extraction of keywords will be performed in order to capture the "true" meaning of the text and then the classification step will take place, to correctly assign the keywords to the **problem**, **solution**, **evaluation** or **results** class. The Keyword Extraction will be performed using the TextRank algorithm, which will be biased using the scores obtained by the TF-IDF algorithm. Once the keywords are obtained, an LSTM Neural Network will be used to classify them as belonging to one of the aforementioned categories. The following sections will provide more details on the different parts that compose the approach, both from a mathematical and algorithmic point of view.

3.1 MATHEMATICAL FRAMEWORK

This section will provide a more detailed description of the algorithms involved in the keyword extraction process, from a mathematical perspective. A fundamental part of the work in this thesis is held by the biasing of the PageRank, reason for which the PageRank itself has been included in this chapter as well.

3.1.1 *The PageRank algorithm*

The PageRank algorithm [21] was originally designed to rank web pages in response to a search query, in order to be able to sort them and return the most relevant ones to the user. In this context it is instead used to rank the words of a text. Every word is given a score recursively based on the score of the words that are linked with it. The basic formula is quite simple and accounts for directed graphs:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} \tag{3.1}$$

where N_v is the number of links from page v, B_u represents the pages that point at page u, and c is a normalization factor. In the undirected graph case, the number of links incoming to is equal to the number of links outgoing from the page. Being a recursive formula, depending

on the configuration of the graph there could be infinite loops, which would generate what is called a *rank sink*. This is for example the case when two pages point at each other but not at any other page, and one of them has an incoming link. This would produce a constant increase in the rank of the two pages, indefinitely. To account for that, a *rank source* vector E is introduced, which allows to simulate a random walk on the graph:

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v} + cE(u) \tag{3.2}$$

The source vector E represents the distribution of the pages in the web, and, as such, the probability for each node to be jumped to in a random walk. In the basic version of PageRank a random walker has equal probability to jump to any node, which leads to $E(u) = 1/N_{nodes}$, $\forall u$, where N_{nodes} is the number of nodes in the graph. By modifying the *rank source* vector E, one could make some nodes more prone to be jumped into than others, making their score higher. This idea is well explained in the original paper [21, section 6] and constitutes the core of the work in this thesis.

3.1.2 *The TextRank algorithm*

TextRank [8] is used as a baseline in almost all the works related to Keyword Extraction, because it is easy to implement and understand, and generates fairly good results. The starting point of the algorithm is the creation of the Graph $G(V, E)$. The vertices represent the words of the text, while the edges represent the relations between the words. Not all the words will be included in the graph, but only adjectives and nouns. Furthermore, an edge is added between two words only if they co-occur within a window of N words in the text. The Graph G, as described in the paper, has been chosen to be undirected and unweighted, although the algorithm can be easily extended to take into consideration both of those cases as well. The core of any Keyword Extraction algorithm is to score and rank the words of the text so that the best ones can be selected. The scoring phase for TextRank is based on the application of PageRank onto the Graph G:

$$S(V_i) = (1 - d) + d \sum_{V_j = Adj(V_i)} \frac{S(V_j)}{|Adj(V_j)|} \tag{3.3}$$

where $d = 0.85$ is a dumping factor, $Adj(V_i)$ represents the list of nodes adjacent to V_i and $|Adj(V_j)|$ represents the number of nodes adjacent to V_j. This formula is a slight variation of the PageRank standard formula (Equation 3.2), and it is run for 20 to 30 iterations or until convergence, with a threshold of difference between the new and the old scores of 0.0001.

Following the scoring phase, the top T words are selected, where T is chosen as one third of the number of words in the text. A post processing phase is then performed to collapse the selected keywords into keyphrases, if they appear consequently in the original text.

3.1.3 *The TF-IDF bias*

The biasing algorithm which has yielded the best results is TF-IDF. As already mentioned briefly in Section 2.1.2, this algorithm is based on the following mathematical formula, which aims at finding a score that represents the importance of word w in document d:

$$w_d = f_{w,d} \log\left(\frac{|D|}{f_{w,D}}\right) \tag{3.4}$$

Taking one component at a time, $f_{w,d}$ indicates the number of times word w appears in document d, $f_{w,D}$ indicates the number of times word w appears in the entire corpus D and $|D|$ indicates the number of terms in the document corpus. The first part, $f_{w,d}$, represents what is called *TF*, or Term Frequency, while the second part of the product, $\log\left(\frac{|D|}{f_{w,D}}\right)$, represents the *IDF*, or Inverse Document Frequency. The *IDF* ensures that if a word with high *TF* appears very frequently in the whole corpus, then its score will be lower than a word that has a high *TF* for the document but is not generally frequent. This simple algorithm is very effective in finding words which are specific for the document and not common to all documents in the corpus. In the proposed solution, the corpus was composed of 1 290 000 documents, for a total of 136 814 223 words.

3.1.4 *Modifications to the original algorithms*

In the proposed solution, some modifications were made with respect to the algorithms as described in the original papers. Concerning the TextRank algorithm, the original paper does not indicate a limit of keywords to collapse when forming the keyphrases, while in this solution, the maximum number of terms to collapse was chosen to be 2. So in case of a sentence such as *"Python code for plotting linear functions"*, if *"plotting"*, *"linear"* and *"functions"* were chosen as keywords, the generated keyphrases would be *"plotting linear"* and *"linear functions"*. Another difference in TextRank, is that in the original paper, after creating the keyphrases, the single words that compose the keyphrase were removed from the list of candidates. In the proposed approach instead, both keyphrases and keywords are kept, and the newly-created keyphrases are scored using the average of the score of the single terms. One final difference concerns the maximum number of iterations for the convergence of the

PageRank. While the TextRank original paper suggests 20 to 30 iterations, in the current work the `max_iter` parameter was set to 100. The remaining parameters were kept as described in the original paper.

3.2 ALGORITHMIC FRAMEWORK

This section will explore the approach described in this chapter from an algorithmic point of view. Both the pseudocode and a summarizing graph of the algorithm will be presented. Listing 3.1 presents the pseudocode for the 5 steps that compose the Keyword Extraction algorithm: "text pre-processing", which aims at lemmatizing the words of the text and assign a part-of-speech tag to each, "graph construction", which deals with the implementation of the graph of word co-occurrence, "creation of bias dictionary", which produces a dictionary of words and their score, based on the bias algorithm selected, "application of biased PageRank", which includes the execution of the biased PageRank algorihm and "post-processing", which aims at creating keyphrases. Listing 3.2 shows the pseudocode for the Classification part, which is composed of two main steps: "Neural Network (NN) input construction", which concerns the creation of the sentences which will be input to the LSTM and "class prediction", which uses the model to predict a class for each input sample. The pseudocode does not include the training of the Neural Network, which must be performed prior to the "class prediction" step, in order to generate the model that allows the classification phase.

Listing 3.1: Keyword Extraction Algorithm

```
INPUT: textual document as text
       window_size as ws
       number of keywords as n_kw
OUTPUT: list of keywords

# step 1: text pre-processing
for word in text:
    apply lemmatization
    apply POS tagger
    if tag=="ADJ" or tag=="NOUN":
        words_of_text.add(word)
generate text_n-grams

# step 2: graph construction
for word1 in words_of_text:
    determine window
```

```
    for word2 in window:
        if word1 in window and word2 in window:
            graph[word1][word2] = 1
    normalize matrix per row

# step 3: creation of bias dictionary
compute TF-IDF score of words_of_text
create bias_dictionary in the format {word: TF-IDF score}

# step 4: biased PageRank
apply biased-pagerank to graph to get words_with_score

# step 5: post-processing
sorted(words_with_score.keys(), key=lambda i: words_of_text.index(i))
form my_n-grams with words_with_score
for ngram in my_n-grams:
    if ngram in text_n-grams:
        ngram_score = np.mean([words_with_score[w] for w in ngram.split(" ")])
        words_with_score[ngram] = ngram_score
return list(words_with_score.keys())[:n_kw]
```

Listing 3.2: Keyword Classification Algorithm

```
INPUT: textual document as text
      list of keywords as keywords
OUTPUT: keywords with classes

#step 1: NN input construction
for kwd in keywords:
    for sentence in text:
        if kwd in sentence:
            add sentence to input_paragraph
tokenize input_paragraph

#step 2: class prediction
feed input_paragraph to NN
return {keyword: class}
```

Figure 3.1 schematically represents the complete algorithm. The two parts are clearly identified, and the steps of which they are composed of are listed in the order of execution. Note that step 2 and step 3 of the Keyword Extraction process are placed side by side, indicating

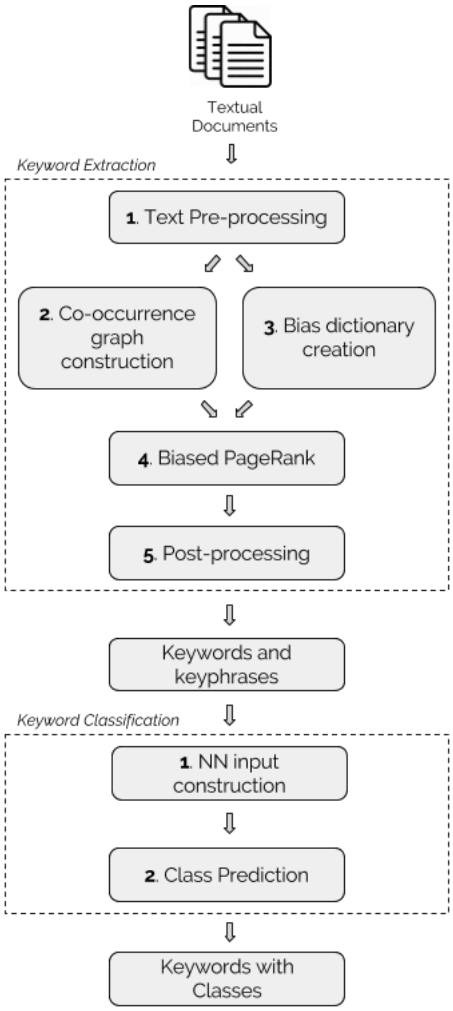

Figure 3.1: Main steps of the complete algorithm

that they could be excecuted in parallel if necessary, since they do not depend on one another. The elements outside the Keyword Extraction and Keyword Classification areas represent the inputs of the following area or the output of the previous one.

4

METHODOLOGY

This chapter aims at describing the performed experiments and the evaluation methodology used in the thesis. In particular, the first sections will present the different datasets used for the experiments and the evaluation metrics adopted. After that, the section describing the baselines, the biasing algorithms and the Neural Network will follow. The chapter will conclude by presenting the experiments and the hypothesis that they aim to prove.

4.1 DATASETS

Three different datasets have been used to conduct different experiments, whose characteristics are summarized in the following table:

Table 4.1: Datasets

Name	*Number of documents*	*Average number of words per document*	*Average number of keywords per document*
Small_full_texts	11	4261	6
IRIS_abstracts	1402	283	7
IRIS_full_texts	60000	3550	6

The *Small_full_texts* dataset includes 11 documents on the topic of Keyword Extraction and Classification. A complete list of the documents included is available in Section A.1. The *IRIS_abstracts* dataset contains abstracts of scientific papers coming from different domains. The biggest dataset is the so-called *IRIS_full_texts*. It is composed of papers coming from the medical domain, and it is constructed to provide access not only to the content and author-assigned keywords, but also to each section of the paper. Its structure is illustrated in Section A.2.

4.2 EVALUATION METRICS

In order to evaluate the results obtained, the extracted keywords were compared with the proposed ones, and precision, recall, and f-measure were computed. It is important to note

that a strict evaluation was performed, which consisted in always extracting the same number of keywords as the ground truth, and performing an exact match between the words. This results in having the same value for precision, recall and, consequently, f-measure. In fact, precision is computed as

$$\text{precision} = \frac{|\text{correctly identified keywords}|}{|\text{keywords extracted}|}$$

while recall is computed as

$$\text{recall} = \frac{|\text{correctly identified keywords}|}{|\text{true keywords}|},$$

which results in the same value for the two denominators, hence for the two measures. Another consequence of the use of a strict evaluation metric is the generally lower number of correctly identified keywords. In fact, variation of the same word, such as *message* and *messages*, are considered to be a mismatch, and so are cases such as *mass media* and *media*. Also, as mentioned in Section 3.1.4, the keyphrases generated by the algorithm are composed of a maximum of 2 terms. Given the strict comparison metric, this means that if the ground truth presented a keyphrase composed of 3 or more terms, there could never be a match. Evaluating the results with this metric allows the user to have an idea of the worst case performances. It is important to keep these considerations in mind when looking at the results, yet knowing that the constraints here applied can be easily softened to obtain better performances.
A different evaluation process is necessary for the classification of the keywords. The output of the LSTM Network is a class that corresponds to a section type (problem, solution, evaluation or results). In order to evaluate the classification obtained, 10 papers were randomly selected and the extraction of keywords was performed. After that, the extracted keywords were manually classified, in order to generate a ground truth, since the classes were not provided with the keywords in the data set. The evaluation of the results was performed separately for the Keyword Extraction and Keyword Classification, using precision, recall and f-measure, and adopting the same strict comparison metric described above.

4.3 THE ALGORITHMS

This section will present a detailed description of all the baseline algorithms and all the biasing algorithms. It is important to note that the TextRank and TF-IDF algorithms have already been addressed in Chapter 3, and are therefore not present in this list. Furthermore, the ExpandRank algorithm has not been used in the final experiments, together with the SingleLinkage bias. ExpandRank and the SingleLinkage algorithms, after the first run, appeared

to be computationally very expensive and not viable for a real life solution therefore were chosen to be left for future experiments.

4.3.1 *Baseline algorithms*

This section will present a list of what have been considered to be the baseline algorithms for the task of Keyword Extraction. A description of the TextRank and TF-IDF algorithms can be found in Section 3.1.2 and Section 3.1.3.

PositionRank

The first baseline is the recently proposed PositionRank algorithm [13]. This algorithm, similarly to TextRank, is based on a slight variation of the PageRank formula (Equation 3.2), here presented in its matrix form:

$$S = \alpha \cdot \tilde{M} \cdot S + (1 - \alpha) \cdot \tilde{p}. \tag{4.1}$$

$\tilde{M}$ represents the co-occurrence normalized matrix. Differently from TextRank, this matrix models a weighted graph, meaning that the entry $\tilde{M}[i][j]$, prior the normalization, is equal to the number of times word i and word j co-occur within a window of a certain size. Continuing with the formula, $\alpha = 0.85$ is the dumping factor and $\tilde{p}$ is the source vector. The difference with PageRank lies precisely in the factor $\tilde{p}$: in order to give different importance to the words based on their position in the text, the elements of the vector $\tilde{p}$ get biased. Instead of giving equal probability to each word, each element of $\tilde{p}$ is calculated as the sum of the inverse of the position of every occurrence of the word in a text. The elements are then normalized as shown:

$$\tilde{p} = \left[\frac{p_1}{p_1 + p_2 + ... + p_{|V|}}, \frac{p_2}{p_1 + p_2 + ... + p_{|V|}}, ..., \frac{p_{|V|}}{p_1 + p_2 + ... + p_{|V|}} \right], \tag{4.2}$$

where, considering $[occ_1, ..., occ_n]$ as the positions of the n occurrences of word i, p_i is computed as

$$p_i = \sum_{x=1}^{n} \frac{1}{occ_x}. \tag{4.3}$$

So if word A is found in position 2, 7 and 39, $p_A = \frac{1}{2} + \frac{1}{7} + \frac{1}{39}$. This allows words that are found at the beginning of the text to have higher probability of being selected as keywords. Once the keywords are extracted, the post-processing phase for this algorithm consists in generating the keyphrases, by concatenating the keywords that appear consequently in the text. In addition to this, the phrases get filtered, in order to match the following regular expression: $(ADJ) * (NOUN)+$, where ADJ indicates an adjective, and NOUN indicates a noun, as per part-of-speech tag.

ExpandRank

The Expand Rank algorithm [30] was chosen to be part of this list because of its two-phase approach to the problem, which constitutes a change to the pattern presented in TextRank and PositionRank. The first phase of the algorithm consists in a neighbor-based analysis, while the second phase is focused on the document level. The idea behind this division resides in the authors' goal to incorporate both local (document-level) and global (neighbor-level) information into the keyword extraction process. During the first phase, the current document d_0, gets *expanded* to a document set D, which contains the top k most similar documents to d_0. The similarity is computed by representing the documents as a vector of terms, each weighted with their TF-IDF score, and by calculating the cosine similarity between these vectors. The distances are also used to weight the contribution of the document: the more distant d_x is from the current document d_0, the less its contribution. The neighbor-level contribution is computed by first constructing a graph $G = (V, E)$ in which the vertices are all the nouns and adjectives of the whole document set D, and the edges are weighted ($aff(w_i, w_j)$) depending on the co-occurrence of the two words within a window of maximum w words:

$$aff(w_i, w_j) = \sum_{d_p \in D} sim_{doc}(d_0, d_p) count_{d_p}(w_i, w_j), \tag{4.4}$$

where $count_{d_p}(w_i, w_j)$ represents the co-occurrence count of word i and word j in document d_p. Onto this graph, the PageRank algorithm [21] is applied to obtain the *saliency scores* for each word, which represent the global information. The second step of the algorithm aims at providing the local information, hence the document-level knowledge. It consists in extracting keyphrases from the current document d_0, by selecting the candidate words found in the previous step and collapsing them into keyphrases if they appear consequently. The score of a keyphrase is given by the sum of saliency scores of the words in the phrase. All the candidate words and phrases from d_0 are ranked according to their score, and the top m are extracted.

Centroid

This method is based on the vector representation of the words in the text. In order to obtain such representation, the `gensim word2vec`[1] model has been used [24]. The result is a unique 100-dimensional vector for each English word contained in the specific dictionary. Given these vectors, it is possible to calculate the centroid word as the average vector. The goal is to compute the distance (here chosen to be euclidean) of every word to the centroid, normalize

1 Documentation available at https://radimrehurek.com/gensim/models/word2vec.html

it ($dist_{norm}$), and score each word with $1 - dist_{norm}$ so that the higher the score, the closer the word is to the centroid. The top T words are then extracted and selected as keywords.

4.3.2 *Biasing algorithms*

This section will present the different algorithms that have been used as biasing criteria for the PageRank algorithm. As stated in Section 1.2, the idea of biasing the PageRank algorithm was inspired by the work in [13], but different experiments were carried out, in order to see if they could lead to better results. The idea is to find different ways to score the words in the text, and then use the normalized scores as source vector E, as described in Equation 3.2. Note that a description of the TF-IDF biasing algorithm has already been addressed in Section 3.1.3.

Entropy Difference

This algorithm [23] provides a way to score the words by analyzing the entropy difference between the intrinsic and the extrinsic modes. For every word, given the position of all its occurrences in the text $(t_1, t_2, ..., t_m)$, the distance between two consecutive occurrences (*arrival time difference*) is calculated as $d_i = t_{i+1} - t_i$. The distance between the last occurrence and the first is taken into account as well, and it is calculated considering the text to be circular by connecting the last and the first word. This distance is therefore computed as $d^*_m = N - t_m + t_1$, where N is the total number of words. At this point, the arrival time difference d_i belongs to the intrinsic mode d^I if $d_i < \mu$, to the extrinsic mode d^E if $d_i > \mu$, where μ is the average of the arrival time differences. Once the intrinsic and extrinsic modes have been computed, the entropy of the two modes can be calculated as follows:

$$\begin{aligned} H(d^I) &= -\sum_{d \in d^I} P_d \log_2 P_d \\ H(d^E) &= -\sum_{d \in d^E} P_d \log_2 P_d \end{aligned} \tag{4.5}$$

where P_d is the probability for the occurrence of microstate d in the system d^I and d^E respectively. The entropy difference can finally be defined as:

$$ED^q(d) = \left(H(d^I)\right)^q - \left(H(d^E)\right)^q \tag{4.6}$$

where $q \in N^+$ is set to be 2 and is used to weigh the entropy difference.
The higher the $ED^q(d)$ score, the higher the probability that word is a keyword. The normalized score is used as source vector E for the PageRank algorithm, following Equation 3.2.

Distance to median and average word vector

The *Centroid* baseline algorithm, as described in the previous section, involves the scoring of the words in a text based on the distance from the average word. This score can be exploited to create the source vector E, described in the PageRank algorithm (Section 3.1.1). If the centroid vector is built as the average vector, this biasing criterion is referred to as *centroid_average*. The centroid vector can also be built as the median vector, in which case this biasing criterion is called *centroid_median*.

Distance to centroid using K-Means and DBSCAN

Both the K-means [31] and DBSCAN [32] clustering algorithms are well known. The main difference between the two is that while K-means forms the clusters based on the distance of the points to the centroid, DBSCAN is a density-based clustering algorithm, which determines the clusters based on the number of points in a specific area. In this work, these algorithms have been used to cluster the vector representations of the words of the text, to be able to use the distance to the centroid as a score for each word, which in turn could be used as a bias for the PageRank algorithm. For the K-means algorithm the number of centroids has been chosen to be equal to $1/10$ of the number of words in the text, and the algorithm has been run by considering the centroids to be the average vector or the median vector. After the clusters and the centroids have been created, the normalized distance ($\mathtt{dist}_{\mathtt{norm}}$) from each word vector to the centroid is computed and once again each word is scored as $1 - \mathtt{dist}_{\mathtt{norm}}$, in order to have the highest score corresponding to the closest word to the centroid. These biasing algorithms are referred to as *kmeans_average* or *kmeans_median*, and *dbscan_average* or *dbscan_median* depending on how the centroid vector is computed (average or median vector).

Distance to centroids using Single Linkage clustering

The single linkage clustering belongs to the category of hierarchical clustering techniques, which can be *agglomerative* or *divisive*. The agglomerative (or bottom-up) clustering algorithms start by considering every data point as a cluster, and then proceed by grouping every time the two closest clusters, while the divisive (or top-down) clustering algorithms start by considering all the data points belonging to one single cluster, and then proceed by dividing it into sub-clusters in order to maximize the dissimilarity between different clusters. Single linkage clustering is an example of agglomerative clustering, in which the distance between two clusters its calculated as the smallest minimum pairwise distance. The `scipy linkage`[2] module has been used for the purpose. After all the possible clusters were computed, the

2 Documentation available at https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html

`sklearn silhouette`[3] score was calculated and used in order to choose the best configuration. Due to its protracted computation time, necessary to compute all the pair-wise distances and the possible clustering configurations, this algorithm has not been included in the experiments.

4.3.3 *Neural Network*

The classification task implemented in this thesis relies on a special type of Recurrent Neural Networks, called Long-Short Term Memory (LSTM). These types of classification models need to be trained, validated and tested. The training process aims at finding the optimal values for the weights that characterize the network, which will ensure that the model is general enough to be able to classify unseen data. In order to achieve that goal, one must properly tune the hyper-parameters of the network, a list of which will be presented in the next section. The next step is to feed the training set to the model so that the weights can be recursively learned. Alongside the training set, a validation set is input as well into the model. The purpose of this validation set is to assess the performances of the training process. In fact, if the model were to be tested on the same data as it was trained, the results would lack generality. The assessment of the training is performed by feeding the validation set into the model and comparing the classification output to the ground truth. In this work, the *accuracy* (i.e. the percentage of correctly identified classes) was used as a metric to evaluate the model. The testing phase is performed separately from the training and validation phase, and aims at verifying the performances on the newly-trained model on unseen data. In this work the training and test data for the LSTM consisted in a set of sentences, obtained as described later in Section 4.5.5.

4.4 EXPERIMENTAL SETUP

The experiments described in this chapter have been run on an Amazon AWS EC2 instance of type m4.4xlarge[4]. It consists of a 2.4 GHz Intel Xeon E5-2676 v3 (Haswell) physical CPU with allocated 6 vCPUs for the machine and 64GB of RAM. The entire code was written in `Python`, v3.4[5]. Table 4.2 presents a list of the main libraries used for the implementation.

3 Documentation available at http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
4 https://aws.amazon.com/ec2/instance-types/
5 https://www.python.org/download/releases/3.4.0/

Table 4.2: List and usage of the main Python libraries

Library name	*Version*	*Usage*
networkx	2.0	Creation of co-occurrence Graph and implementation of the PageRank algorithm
gensim	2.3.0	Word2Vec model
Keras	2.1.2	Text tokenization and NN layers construction
tensorflow	1.4.1	Backend for Keras

4.5 EXPERIMENTS

This section will describe the experiments that have been conducted, starting from those concerning the Keyword Extraction task and concluding with those related to the Neural Network and the Keyword Classification.

4.5.1 *Starting hypothesis*

For the Keyword Extraction task, the core of the algorithm consists in using a biased version of TextRank to try and achieve comparable results to PositionRank, which utilizes a position-based bias. The reason for the following experiments lies in the fact that the premises of PositionRank might not be true and as general as the authors make them sound. Florescu et al. base their idea on the hypothesis that the sooner a word appears in a text, the more important that word is for the text, hence the higher the chance that word will be a keyword. When texts present an abstract at the beginning, this hypothesis holds pretty well, and PositionRank actually outperforms all the other algorithms. This is not surprising if one thinks about what is the role of the abstract in a paper: it represents a summary of the content, hence providing an overview on all the main concepts of the paper. Therefore, when extracting few, general keywords, it is highly probable they are going to be found at the beginning of the document. But what would happen if the document did not contain an abstract? Or what if one was interested in extracting more specific keywords that described all the different parts of the paper, not only the general concepts? These are the questions that drove the work of the first part of the thesis. The solution adopted aims at providing an alternative to PositionRank, whenever its limitations prevent it from obtaining outstanding performances. To achieve this goal, the chosen biases are not based on the position of the word anymore, but rather, for example, on the entropy difference between the intrinsic and extrinsic modes, or on the TF-IDF scores.

4.5.2 *Baseline choice*

The first experiment aimed at determining which baseline was performing the best, so that it could be used as the basic algorithm upon which to build the bias. The 4 selected baseline algorithms are the following:

- TextRank
- PositionRank
- TF-IDF
- Centroid

For a detailed description the reader can refer to Section 4.3.1. As shown in the plots in Chapter 5, three different window sizes were addressed: 3, 5, and 10. The window size concerns only the two graph-based baselines, TextRank and PositionRank, since the other baseline algorithms are not affected by this parameter. The size of the window, for every word w in the text, indicates how many words before and after w to consider in order to determine the co-occurrence of w with the other words. In practice, the bigger the window size, the more connections will be present in the graph. The experiment was conducted on the *IRIS_abstracts* dataset, and it is easy to observe that the overall best was achieved by TextRank, which was therefore chosen as basic algorithm on top of which to try the different biases.

4.5.3 *Bias analysis*

The next experiments involved applying the biasing of the PageRank on top of the selected TextRank algorithm. The first experiment consisted in applying the biased-TextRank on the *IRIS_abstracts* dataset, in order to have a general idea of the performances. Nine were the algorithms used as a bias for the TextRank:

- centroid_average
- centroid_median
- dbscan_average
- dbscan_median
- entropy_difference
- kmeans_average

- kmeans_median
- position
- tf-idf

All the algorithms have been already described in details in Section 3.1.3 and Section 4.3.2. The 'position' bias refers to the same calculation proposed in the PositionRank algorithm, though this time applied on top of the TextRank. TextRank and PositionRank, although being very similar to one another, present a difference both in the graph construction and in the computation of the keyphrases, as explained in Section 4.3.1, which may lead to different results and was therefore worth observing. The results in terms of precision, recall and f-measure were compared with the 4 baselines, with 3 different window sizes: 3, 5 and 10. Given the promising results, as a follow-up, a second experiment was conducted on the *Small_full_texts* dataset, to see how the algorithms would perform in a more realistic scenario. The results can be seen in Figure 5.3. The third experiment involved a variation of the *Small_full_texts* dataset, in order to verify the performances of the algorithms in a scenario in which the texts do not present an abstract. Again the same conditions in terms of chosen biases and window sizes were kept (results shown in Figure 5.5). The fourth and final experiment involved the modification of the ground truth keywords, which were chosen to be less general and cover more details of each paper (results shown in Figure 5.6). As a consequence, the average number of keywords per document raised to 15 (compared to the previous one which was 6, as shown in Table 4.1).

4.5.4 *Neural Network parameter tuning*

Following the extraction of keywords, the second part of the experiments concerned their classification. As mentioned in the previous chapter, an LSTM Neural Network was chosen and built in order to complete the task. These kind of Networks present some parameters that have to be tuned and properly set in order to determine the configuration that yields the best results. This is the aim of the following experiments. The targeted parameters and their values are summarized in Table 4.3. To conduct the experiment, all the combinations of these values have been tested, and the setting that reached the best result in terms of validation accuracy was selected. The chosen values for each parameter are indicated in bold in Table 4.3.

4.5.5 *Training the Neural Network*

The training of the Neural Network was performed using the *IRIS_full_texts* dataset. As stated in Section 4.1 and shown in Appendix A, this dataset provides information about the differ-

Table 4.3: Neural Network parameter names and tested values. In bold the chosen values.

Parameter	*Testing Values*
Number of neurons for LSTM layer	**128**
Dropout Ratio	**0.2**
Recurrent Droput Ratio	**0.2**
Activation Function	**Sigmoid**
Loss Function	**Categorical Crossentropy**
Optimizer	**Adam**
Batch Size	**128**
Number of epochs	**15**, 30

ent sections of each document, specifying their title and content. This was essential for the training of the NN, since the goal is to classify keywords as belonging to a specific section type.

Table 4.4: Section Types and Characterizing Words

Section Type	*Characterizing Words*
Problem	introduction, background, problem definition, problem formulation, problem area
Solution	implementation, method, model, approach, solution, algorithm, overview, system, contribution, design
Evaluation	evaluation, metrics, discussion, analysis, environment
Results	results, experiments, conclusion

Therefore, the dataset was filtered in order to determine which documents contained both keywords (necessary to evaluate the Keyword Extraction) and sections that corresponded to either introduction, implementation, evaluation or solution. In order to do so, 3 to 10 words which characterize the 4 types of sections were selected, as shown in Table 4.4. The sections whose title contained one of the characterizing words were then used to build a training set for the Network. The final section set contained 27450 sections of type "introduction/problem definition", 5608 sections of type "solution", 24930 sections of type "evaluation" and 24391 sections of type "results". The input data to the LSTM was constructed as follows, considering one section type at the time. (I) The set of unique words included in the section type was identified. (II) For each unique word, the set of sentences that contain that word was identified.

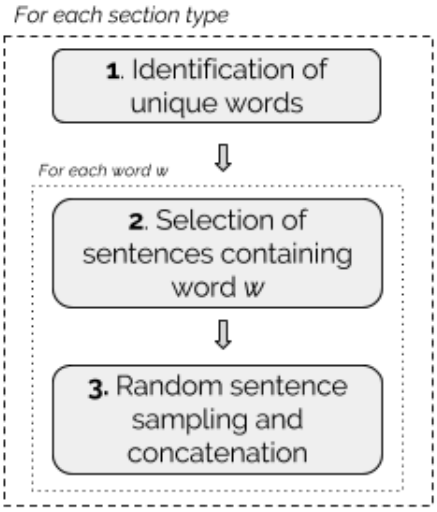

Figure 4.1: Creation of LSTM input

(III) Among those a sample of 5 sentences[6] was extracted and concatenated to form a training sample. The process is repeated for every section type. The final set of samples was then shuffled, and 80% was included in the training set, while the remaining 20% was saved and kept for the validation of the model. Figure 4.1 schematizes the steps here described.

4.5.6 *Experiment on Keyword Classification*

The final experiment consisted in testing the combination of the two parts that compose the algorithm presented in this thesis, in order to extract keywords and evaluate the performances on the classification. In order to do so, a subset of the *IRIS_full_texts* dataset was saved and used as input data. The data was obtained by filtering the documents and keeping only those that presented at least one section per each class (problem, solution, evaluation and results) and at the same time presented ground truth keywords. The final test set included 3985 files.

6 In case of a number of sentences lower than 5, all of the sentences available were selected.

5

RESULTS

This chapter will present the results obtained from the different experiments. Figure 5.1 shows the comparison of the 4 baseline algorithms (TextRank, PositionRank, TF-IDF and Centroid) applied to the *IRIS_abstracts* dataset. On the x-axis 3 different window sizes are indicated, and for each of them, on the y-axes the values of precision, recall, and f-measure are shown. It is important to note that the TF-IDF and Centroid approaches do not involve a window size parameter, explaining why their results are constant for all window sizes. The graph clearly shows that TextRank is outperforming all the other algorithms, performing more than twice as good as the Centroid approach, and presenting a margin of up to 4% with respect to the PositionRank algorithm. It is also noticeable that the a bigger window size seems to be beneficial for PositionRank, but yields worse results for the TextRank. Figure 5.2 shows the same 4 base-

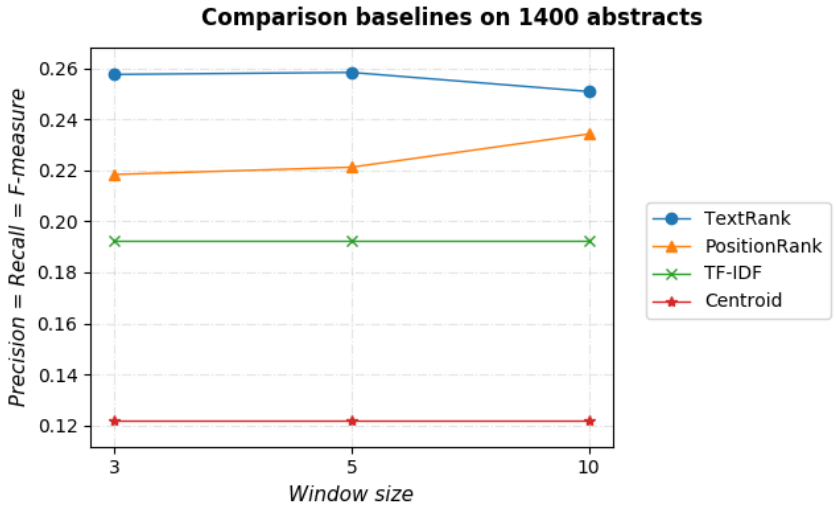

Figure 5.1: Comparison of baseline algorithms on *IRIS_abstracts* dataset. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2.

lines, compared with the biased-TextRank on the *IRIS_abstracts* dataset. For each of the 3 explored window sizes, on the x-axis the 9 different biases are presented, sorted from left to right in increasing performance order. On the y-axes the levels of precision, recall and f-measure are indicated. As shown in the graph, the TF-IDF-biased TextRank is able to outperform both the TextRank itself (by a margin of 2.5%) and all the other baselines. When applying the

other biasing algorithms, the biased TextRank manages to perform better than PositionRank, TF-IDF and the Centroid approach, except with the usage of the "entropy_difference" and "position" biases. The same graph structure has been used in Figure 5.3, which presents the

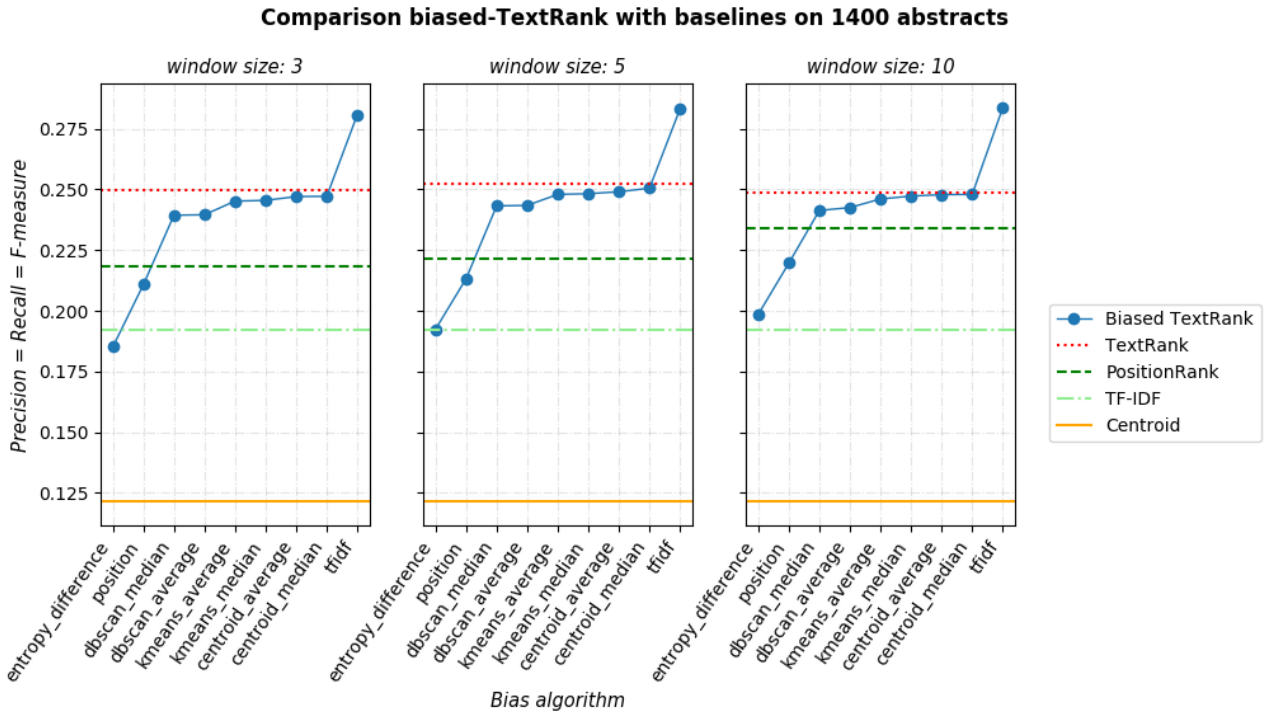

Figure 5.2: Comparison of the biased TextRank with the baselines on abstracts. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2.

comparison of the biased-TextRank with the baselines on the *Small_full_texts* dataset. The first thing that can be noticed from the graph is that the PositionRank is now the leading algorithm, with peaks of almost 25% f-measure. The position-biased TextRank is though able to meet the PositionRank's performances in the case of a window size equal to 5. The entropy_difference-biased and position-biased TextRank are able to outperform the TextRank baseline and the other biased versions of the algorithm, by a margin of at least 4%. Figure 5.4 presents instead a comparison of the performance of PositionRank, in 3 different scenarios. The first (blue) line shows the performance in terms of precision, recall and f-measure when the algorithm is applied on full texts. The second (orange) line concerns the application of PositionRank on texts that do not present an abstract. The last (green) line shows the performance in case the ground truth is composed of more specific keywords. Figure 5.5 shows the comparison of the biased-TextRank with the baselines on a modified version of the *Small_full_texts* dataset, in which the abstracts have been removed from each text. The PositionRank algorithm and

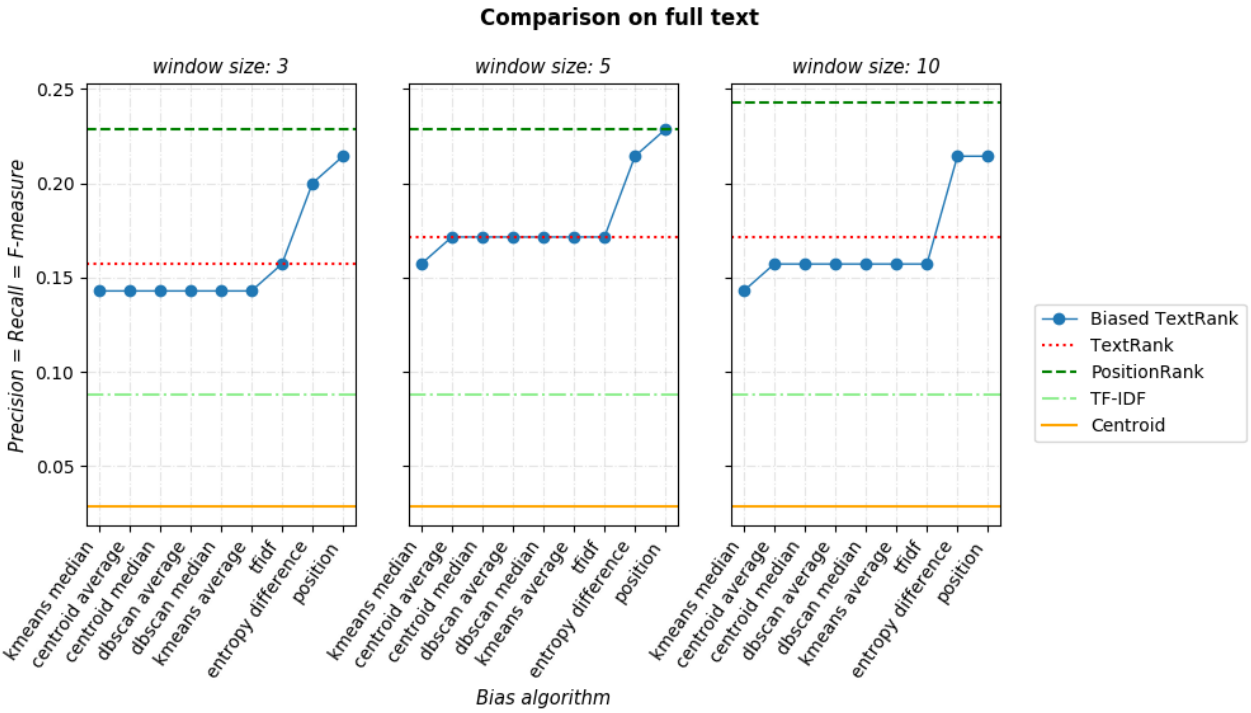

Figure 5.3: Comparison of the biased TextRank with the baselines on full texts. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2.

the position-biased TextRank obtain once again the best results, this time with peaks as high as 22.5% f-measure. All the versions of biased-TextRank are also able to at least meet the results of the TextRank baseline, while the TF-IDF and Centroid baselines remain the worst-performing algorithms. Figure 5.6 represents the same comparison between biased-TextRank and baseline algorithms on the *Small_full_texts* dataset with modified ground truth keywords, which have been chosen to be more specific and less general. By looking at the graph, one can notice that the "position" bias is not anymore the leading biasing algorithm, replaced by the "TF-IDF" bias, which outperforms it by a margin of 2%. The remaining biases perform similarly to the TextRank baseline, except for the "entropy_difference" biasing algorithm, which outperforms TextRank by a margin of up to 5%. The PositionRank baseline still obtains the best results, with peaks of more than 22.5% f-measure.

The experiments regarding the Keyword Classification task concern the training of the LSTM model and the evaluation of the classification of the extracted keywords. The training of the Neural Network, with the parameters indicated in bold in Table 4.3, led to a validation accuracy of 54%. The model was then used to classify the keywords extracted by the TF-IDF-biased

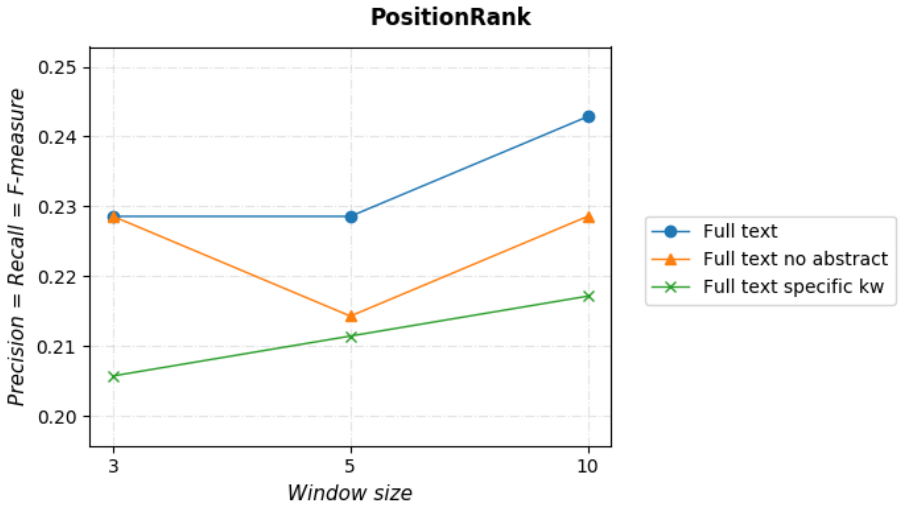

Figure 5.4: Performance of the PositionRank algorithm in case of full texts, full texts with no abstract, and in case the ground truth keywords were chosen to be less generic. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2.

TextRank, and the results were compared to the manually-generated ground truth. Both the predicted and the ground truth classes are shown in Table B.3 and Table B.4. The classification task obtains 22% in precision, recall and f-measure.

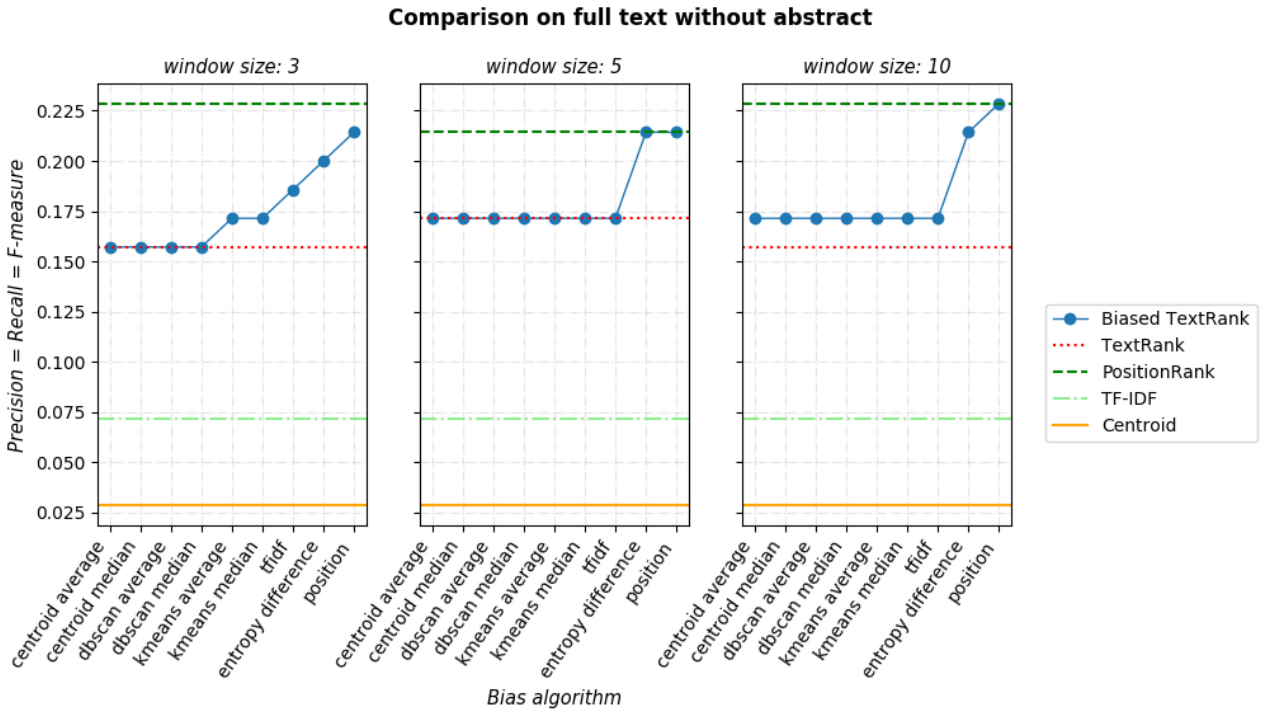

Figure 5.5: Comparison of the biased TextRank with the baselines on full text after the removal of the abstract. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2.

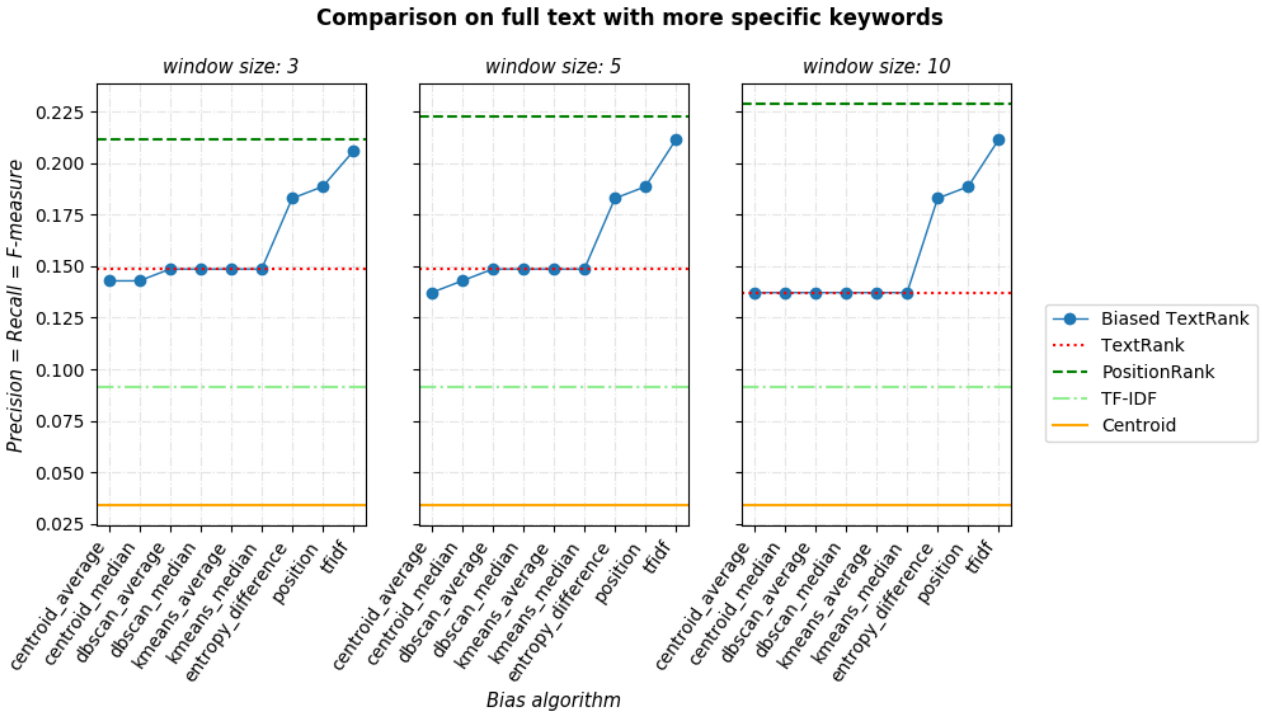

Figure 5.6: Comparison of the biased TextRank with the baselines on full texts, modifying the ground truth keywords to be less general. The y-axis represents the value of precision, recall, and f-measure, calculated as described in Section 4.2.

6 COMMENTS AND DISCUSSION

6.1 BIASED PAGERANK: THE REASON BEHIND IT

The idea of a personalized PageRank had been introduced already in the original paper [21], in 1998, but it wasn't until recently that Florescu et al. applied it to the process of Keyword Extraction [13]. Inspired by their work, the biased PageRank has been the central idea of the first part of this thesis. The idea behind it is that the biasing of the PageRank algorithm provides some additional knowledge to the overall algorithm, which will help obtaining better quality keywords. Each bias gives a different contribution, and has the potential to provide better results for the Keyword Extraction. For example, using the TF-IDF as bias should drive the choice to those words which are characteristic of the document, those that best help to distinguish it from others. On the other hand, the approaches that rely on clustering, the idea is that each cluster would capture one of the aspects of the text, and the centroid words and the ones closer to them would be a good representation of these concepts. When looking at the position-based biasing, the words at the beginning of the text will have a higher importance than the ones found towards the end, making the latter less likely to be selected as keywords. When talking about using the distance to the central word as bias, the idea is that the central word should capture the overall meaning of the document, and the closer a word is to it, the better it is. Last but not least, by calculating the entropy difference between the intrinsic and the extrinsic modes, the words that are equally spread in the text would get a lower score than those that are found only in certain areas of the document. This reflects the idea that keywords tend to attract each other, while common words are typically uniformly spread within a document.

6.2 ANALYSIS OF THE RESULTS FOR KEYWORD EXTRACTION

This section will analyze and explain the results shown in Chapter 5. Figure 5.1 represents the first experiment, which aimed at understanding which baseline performed the best. It is immediate to see that the TextRank algorithm performed better than the other candidates, regardless of the window size used. It was therefore chosen as the designated baseline to perform the next experiments. Once the base algorithm was identified, the next phase involved the application the biases on top of the TextRank to determine which of those was perform-

ing the best. The results in Figure 5.2 show that the biasing of the PageRank on top of the TextRank was able to outperform the baselines in multiple occasions, reaching peaks of more than 27% F-measure. It is also noticeable how the newly-proposed *centroid* method, when used as a bias on top of the TextRank, obtains comparable results to the baseline, proving the validity of the algorithm. Given the promising results, as a follow-up, a second experiment was conducted on the *Small_full_texts* dataset, to see how the algorithms would perform in a more realistic scenario. In this case though, the results show that the PositionRank and the position-biased TextRank obtain the best results (Figure 5.3). This is, however, not surprising, since all the papers included in the dataset presented an abstract section at the beginning of the document. In fact, as briefly mentioned in Section 4.5.1, the abstract represents a summary of the content of the document, and as such, the words contained in it represent the perfect candidate keywords. The reason why the PositionRank algorithm and the position-biased TextRank are able to capture these words, highly depends on their location in the document. But if the summary was not included, the PositionRank should not be performing as well as it did. Similarly, if the ground truth keywords were not chosen to be general, but instead selected to capture more detailed aspects of the document, the performances of the PositionRank should decrease. To verify these assumptions, different experiments were performed. First, the performance of the PositionRank were calculated in 3 different scenarios: when the algorithms was applied to full texts, when it was applied to full texts without abstracts, and when the ground truth was modified so that the keywords were less general. As expected, the performances in the two latter cases decrease compared to the case in which PositionRank is applied to full texts (Figure 5.4). PositionRank was then compared to the other baselines and to the biased-TextRank, on the *Small_full_texts* dataset, after removing the abstract from the documents. As shown in Figure 5.5, despite the drop in performances of the PositionRank, it still represent the leading algorithm, although now the biased TextRank is able to meet the PositionRank's performances in 3 occasions. The final experiment consisted in changing the ground truth keywords. The newly selected words are more specific than the previous ones, as shown in the example provided in Table 6.1. In the table, the left column presents the keywords assigned by the authors of *Keyword extraction by entropy difference between the intrinsic and extrinsic mode* [23] are shown, while on the right column the list gets expanded in order to give a more thorough picture of the content of the document. The different algorithms were ran again on the *Small_full_texts* dataset, with the new set of ground truth keywords. The results in Figure 5.6 show a considerable drop in performance of the position-biased TextRank, of up to 4% in F-measure score, which caused this bias to lose its leading position. This shows how the presence of the bias is able to influence the basic algorithm it is applied to, confirming the good intuition behind the work.

Table 6.1: Example of modified list of keywords on *Small_full_texts* dataset

Normal Keywords	***Specific Keywords***
keyword extraction entropy difference intrinsic mode extrinsic mode	keyword extraction ranking statistical analysis spectra entropy difference intrinsic mode extrinsic mode occurrence frequency distribution topic local separation microstate entropy word word token word type term book boundary condition statistical distribution

It is important to keep in mind that the results were obtained using a strict evaluation metric, which puts hard constraints on what is considered to be a match with the ground truth, hence presenting the performances in the worst case. For example, words like *email* and *e-mail* are considered to be a mismatch. In the future, some pre-processing of the ground truth might be needed to fix this imperfection of the metric used. It would be quite easy to relax these constraint to obtain better results in terms of precision, recall and f-measure, for example by considering as a match the cases in which a selected keyword is included in one of the ground truth keywords, or allowing two keywords to match if they share the same origin of the word. An example of the changes in performances when relaxing the constraint is available in Table 6.2. Another factor that could influence the results is the subjectivity of the keyword selection task. As discussed in Section 1.1, when two people are given the same text, it is highly probable that the keywords selected by the first person are going to be different from

Table 6.2: Difference in F-measure when relaxing constraints on the comparison of keywords. Matching keywords indicated in bold.

True Keywords	*Selected Keywords*	*Selected keywords (Relaxed metric)*
infancy	infant	**infant**
medium	**medium**	**medium**
education	study	study
mass	mass medium	**mass medium**
exposure	**exposure**	**exposure**
empowerment	regional	regional
socio	mother	mother
economic	regional variation	regional variation
mortality	infant mortality	**infant mortality**
F-measure with strict metric: 22%		
F-measure with relaxed metric: 55%		

the ones selected by the second person. The key point here is that the two sets of keywords could be both equally good, therefore being different doesn't necessarily mean that one of the two sets is wrong. On this regard, sometimes texts yielding higher f-measure (30%) and texts yielding lower f-measure (0%) when double-checked by human might be indistinguishable in the sense that the keywords provided by the algorithm in both cases make equal sense and seem to capture the "true" meaning of the texts (examples can be found in Table B.2 and Table B.1). Cases like this highlight the difficulty of this task, particularly related to its evaluation. A different element influencing the results in this thesis is the choice of keeping the maximum number of terms in a keyphrase equal to 2. As already mentioned in Section 4.2, having this limit in combination with the strict comparison metric yields generally lower performances. The reason why it was set to this value is due to the keywords in the *IRIS_abstracts* dataset, which were for the most part composed of maximum two terms. When moving to the *IRIS_full_texts* dataset though, the keywords seemed to be mostly composed of three or more terms. Due to time constraints, it was not possible to investigate this further to determine the best value for the parameter, which was left as it was. One final remark concerns some of the biasing algorithms which have not performed as well as the others. In particular, the DBScan and Kmeans biases generally present the lowest performances. Although the two methods have potential, they haven't been explored in much details, to allow focusing on more detailed experiments around the other more promising biases. In fact, the tuning of the

DBScan and Kmeans parameters allows for a lot of possible outcomes, which haven't been addressed in this work, hence leaving space for further work and improvement.

6.3 ANALYSIS OF THE RESULTS FOR THE KEYWORD CLASSIFICATION

This section will discuss the results obtained when classifying keywords, during the final experiment on the *IRIS_full_texts* dataset. The system takes as input a subset of 3985 textual documents, and outputs for each document a list of keywords and the class each belongs to. As discussed in Section 4.2, 10 papers were randomly extracted, and the extracted keywords were manually classified to generate the ground truth. The complete list of papers and extracted keywords is available in Table B.3 and Table B.4. The first version of the classifier, in terms of precision, recall and f-measure, obtained 22%. As can be seen from the detailed results though, the output of the classifier appeared to be strongly biased towards the section of type "Evaluation". The reason is to be found in the lack of time, necessary to explore and tune all the hyper-parameters of the model. The constructed LSTM is though a good starting point and with proper setting of the parameters and consequent re-training of the model, the results would improve and the system would be able to show its full potential.

6.4 THREATS TO VALIDITY

This section aims at presenting the limitation of this work, which could compromise its validity. In particular, as mentioned in Section 4.1, the dataset used to conduct the final experiment included papers belonging solely to the medical domain, which limits the validity of the results on that area only. Moreover, the experiments on Keyword Extraction performed on full texts were run on the *Small_full_texts* dataset, composed of only 11 papers. For the results to be generally valid, the experiments need to be executed on a bigger dataset. In addition to this, the *IRIS_full_texts* dataset contained a considerable number of ground truth keyphrases which were composed of more than 2 terms, which meant that they could never be a match with the extracted ones. Also, some of the ground truth keywords were not present in the original document, meaning that the authors were rephrasing the main concepts, rather than extracting them directly from the text. Again, this means that those keywords were automatically a *miss* when computing precision, recall and f-measure. Given more time, the fixing of these two aspects would be possible and the performances would consequently increase. Finally, the tuning of the parameters of the Neural Network could not be thoroughly explored, due to time constraints concerning the thesis work, which led to the best model having a strong bias towards one class. Table 4.3 shows the parameters that were chosen to create

the LSTM model, which were, for the vast majority, left to the default values[1]. The process for fixing this problem consists in the proper tuning of the hyper-parameters of the model. Although straightforward, it could be time consuming.

1 Default values taken from: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

7

FINAL REMARKS AND FUTURE WORK

7.1 CONCLUSIONS

The work presented in this thesis focused on providing an efficient and effective way to extract and classify keywords. The presence of keywords, in fact, would significantly speed up the automatic understanding of the content of a document, while their classification would provide some more insights on the structure of the document itself. In order to achieve this goal, inspired by the work of Florescu et al. in PositionRank [13], who use the position of the words in the text to bias the score of the words, a different version of the TextRank algorithm was proposed, which was biased using the TF-IDF score instead of the position. The results are particularly good in the case of application of the algorithm to abstracts, in which the TF-IDF-biased TextRank was by far the best performing algorithm. In the other cases, the proposed solution is able to outperform the TextRank, TF-IDF and Centroid baselines in all the experiments, and match the results of PositionRank in two occasions. For the classification part, an LSTM Neural Network was adopted as a classifier to assign the extracted keywords to the correct category: **problem**, **solution**, **evaluation** or **results**. Despite the promising outcomes of the algorithm, the time frame of the thesis did not allow for a thorough examination of all the possibilities, and some details have remained unexplored. The following section aims at presenting some suggestions on how to continue and expand the work of this thesis.

7.2 FUTURE WORK

Different directions can be taken to improve, complete and expand the work of this thesis. One could start by looking at the baseline algorithms, or focus on the biases, or work on the classification task. Concerning the baselines, it was mentioned that the ExpandRank algorithm was not included in the experiments due to its computational time. An optimization of the code could be performed, in order to include it and have a more thorough comparison of the proposed approach with existing algorithms. As for the biases, it would be interesting to explore in more details the newly-proposed centroid approach. In particular, one could start by analyzing why, despite the results of the TextRank and Centroid baseline present a gap in performances of around 13%, when the centroid algorithm was applied as a bias the results were comparable to those of the TextRank, if not better. This could lead to a broader

analysis of the biases, for example trying to understand in what part does the bias influence the results, and why. Still concerning the biases, one could think about performing a fine tuning of the k-means and DBscan algorithms, in order to see if they could lead to better results, or implementing the SingleLinkage algorithm in a more efficient way, so that it could be included in the experiments as well. As future work, it could be possible to find a way to automate the evaluation process for the classification part, which was manually performed in this thesis. Regarding the classification part, the most significant contribution to this work would be to automate the evaluation process for the keyword classification task. This would allow the execution of the algorithm on a bigger dataset and a more precise measure of the performances. Another possibility consists in thoroughly tune the hyperparameters of the LSTM model, in order to be able to choose the best configuration. Finally, on a more general note, it could be interesting to try to optimize the code and make some considerations and comparisons about the time complexity of the algorithm.

APPENDIX A: DATASETS

A.1 DOCUMENTS IN *small_full_texts* DATASET

1. *Keyword extraction by entropy difference between the intrinsic and extrinsic mode* [23]
2. *Single Document Keyphrase Extraction Using Neighborhood Knowledge* [30]
3. *KEA: Practical Automatic Keyphrase Extraction* [19]
4. *Keyword and Keyphrase Extraction Techniques: A Literature Review* [12]
5. *Keyword Extraction from Emails* [9]
6. *KX: A flexible system for Keyphrase eXtraction* [18]
7. *The PageRank Citation Ranking: Bringing Order to the Web* [21]
8. *PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents* [13]
9. *Automatic keyword extraction from individual documents* [22]
10. *TextRank: Bringing Order into Texts* [8]
11. *Using TF-IDF to Determine Word Relevance in Document Queries* [17]

A.2 *iris_full_texts* DATASET FORMAT

The following code illustrates the format (JSON) of the files contained in the *IRIS_full_texts* dataset. Squared brackets "[]" indicate a list, while curly brackets "{}" indicate a dictionary.

Listing A.1: Format of *IRIS_full_texts* dataset

```
{
    "pmcid": <document id>,
    "doi": <doi>,
    "abstract": <abstract section>,
    "author_info":
```

```
    [
        {
            "affiliation": <affiliation author 1>,
            "name": <name author 1>
        },
        ...
        {
            "affiliation": <affiliation author M>,
            "name": <name author M>
        }
    ],
    "language": "en",
    "author_notes": <notes of the author>,
    "keywords": [<list of keywords>],
    "fulltext":
    [
        {
            "section_title": <title section 1>,
            "section_text": <text section 1>
        },
        ...
        {
            "section_title": <title section N>,
            "section_text": <text section N>
        },
    ],
    "title": <title of the document>
}
```

A.3 DOCUMENTS FOR LSTM EVALUATION

For the evaluation of the performances of the LSTM, a sample of 10 random documents was extracted and the keywords provided were manually classified. The following list indicates the PMCID and title of the papers utilized for the evaluation.

1. PMC4613277 - *An Ensemble Method to Distinguish Bacteriophage Virion from Non-Virion Proteins Based on Protein Sequence Characteristics*
2. PMC4970041 - *A Three-Dimensional Shape-Based Force and Stiffness-Sensing Platform for Tendon-Driven Catheters*

3. PMC4447720 - *A framework to observe and evaluate the sustainability of human–natural systems in a complex dynamic context*
4. PMC4072843 - *Rapid molecular genetic diagnosis of hypertrophic cardiomyopathy by semiconductor sequencing*
5. PMC4505166 - *Transcriptome and ultrastructural changes in dystrophic Epidermolysis bullosa resemble skin aging*
6. PMC5676814 - *Addressing sufciency of the CB1 receptor for endocannabinoidmediated functions through conditional genetic rescue in forebrain GABAergic neurons*
7. PMC3897982 - *A systematic review and meta-analysis of the effects of antibiotic consumption on antibiotic resistance*
8. PMC4741544 - *Mitochondrial mass, a new metabolic biomarker for stem-like cancer cells: Understanding WNT/FGF-driven anabolic signaling*
9. PMC1173079 - *Patient-initiated switching between private and public inpatient hospitalisation in Western Australia 1980 – 2001: An analysis using linked data*
10. PMC5527246 - *Development and Optimization of a New Chemoenzymatic Approach for the Synthesis of Peracetylated Lactosamine (Intermediate for the Synthesis of Pharmacologically Active Compounds) Monitored by RP- HPLC Method*

B

APPENDIX B: RESULTS

B.1 EVALUATION OF RESULTS: BEYOND THE NUMBERS

This section aims at showing how, when performing the task of Keyword Extraction, the numbers in the results are often misleading. In particular, Table B.1 shows a sample text and the extracted keywords, where the results show a 30% in f-measure, while Table B.2 shows another text and keywords, with a resulting f-measure of 0%. If one stops at the results, it might seem that the keywords for the first text were better than the keywords for the second text. Looking at them closely though, one could argue that they actually describe the document pretty well, capturing its "true" meaning.

Table B.1: Example of Keyword Extraction performed with the TF-IDF biased TextRank on a sample text taken from *IRIS_abstracts* dataset. F-measure: 30%. In bold the extracted keywords, at the bottom the true keywords. The correctly identified keywords are underlined.

In this paper we first **describe** the class of log-Gaussian **Cox processes** (LGCPs) as models for **spatial** and spatio-temporal point **process** data. We discuss **inference**, with a **particular** focus on the computational challenges of likelihood-based **inference**. We then demonstrate the usefulness of the LGCP by **describing** four applications: estimating the intensity surface of a **spatial** point **process**; investigating **spatial** segregation in a **multi-type process**; constructing **spatially** continuous maps of disease risk from **spatially discrete** data; and real-time health surveillance. We argue that problems of this kind fit naturally into the realm of **geostatistics**, which traditionally is defined as the study of **spatially** continuous processes using **spatially discrete** observations at a finite number of locations. We suggest that a more useful definition of **geostatistics** is by the class of scientific problems that it addresses, rather than by particular models or data formats.
True Keywords: temporal, MCMC, cox process, hierarchical model, continuous, spatial, MAUP, count data, INLA, geostatistics

B.2 RESULTS FOR THE CLASSIFICATION TASK

Table B.2: Example of Keyword Extraction performed with the TF-IDF biased TextRank on a sample text taken from *IRIS_abstracts* dataset. F-measure: 0% In bold the extracted keywords, at the bottom the true keywords.

We propose an analytical framework for **studying bidding behavior** in **online auctions**. The framework focuses on three key dimensions: the multi-stage process, the types of value-signals employed at each phase, and the dynamics of **bidding behavior** whereby early choices impact subsequent **bidding decisions**. We outline a series of propositions relating to the **auction** entry **decision**, **bidding decisions** during the **auction**, and **bidding behavior** at the end of an auction. In addition, we present the results of three preliminary field studies that investigate factors that influence consumers' value assessments and **bidding decisions**. In **particular**, (a) due to a focus on the narrow **auction** context, consumers under-search and, consequently, overpay for widely available commodities (CDs, DVDs) and (b) higher **auction** starting prices tend to lead to higher winning bids, particularly when comparable items are not available in the immediate context. We discuss the implications of this research with respect to our understanding of the key determinants of consumer behavior in this increasingly important arena of purchase **decisions**.
True Keywords: dimension, analytical, process, framework, assessment, particularly

Table B.3: Detailed results for the classification task (1/2). P: Problem, S: Solution, E: Evaluation, R: Results.

	Document PMCID	*Extracted Keywords*	*True Class*	*Output Class*
1.	PMC4613277	virion protein	P	E
		sequence	P	E
		prediction	S	E
		protein	E	E
2.	PMC4970041	catheter	P	E
		force	S	E
		force stiffness	E	E
		stiffness	E	E
		estimation	E	E
		tip	S	E
3.	PMC4447720	sustainability	S	E
		et	P	E
		dimension	S	E
		boundary	E	E
		knowledge	S	E
		new	S	E
4.	PMC4072843	hcm	P	E
		sequence	S	E
		sanger sequence	E	E
		cardiomyopathy hcm	P	E
5.	PMC4505166	age	P	E
		skin	P	E
		rdeb	p	E
		gene	R	E
		protein	E	E
		age process	P	E

Table B.4: Detailed results for the classification task (2/2). P: Problem, S: Solution, E: Evaluation, R: Results.

	Document PMCID	*Extracted Keywords*	*True Class*	*Output Class*
6.	PMC5676814	cb1	P	E
		et	P	E
		cb1 receptor	P	E
		receptor	P	E
		gabaergic	R	E
7.	PMC3897982	resistance	P	E
		journal	P	E
		antibiotic	P	E
		infectious	S	E
8.	PMC4741544	cell	S	E
		fgf3	E	E
		wnt1	S	E
		isogenic cell	S	E
		cell model	S	E
9.	PMC1173079	public	E	E
		couplet	S	E
		phi	R	E
		interval	S	E
10.	PMC5527246	product	R	E
		hydrolysis	E	E
		synthesis	E	E
		oligosaccharide	R	E
		substrate	R	E
		yield	R	E

BIBLIOGRAPHY

[1] Wesley W. Chu, Victor Zhenyu Liu, and Wenlei Mao. "Techniques for Textual Document Indexing and Retrieval via Knowledge Sources and Data Mining." In: *Clustering and Information Retrieval*. Boston, MA: Springer US, 2004, pp. 135–159. ISBN: 978-1-4613-0227-8. DOI: 10.1007/978-1-4613-0227-8_5. URL: https://doi.org/10.1007/978-1-4613-0227-8_5.

[2] Jamie Callan. "Document Filtering with Inference Networks." In: *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '96. Zurich, Switzerland: ACM, 1996, pp. 262–269. ISBN: 0-89791-792-8. DOI: 10.1145/243199.243273. URL: http://doi.acm.org/10.1145/243199.243273.

[3] Vishal Gupta and Gurpreet Lehal. "A Survey of Text Summarization Extractive Techniques." In: 2 (Aug. 2010).

[4] Fabrizio Sebastiani. "Machine Learning in Automated Text Categorization." In: *ACM Comput. Surv.* 34.1 (Mar. 2002), pp. 1–47. ISSN: 0360-0300. DOI: 10.1145/505282.505283. URL: http://doi.acm.org/10.1145/505282.505283.

[5] Aniket Rangrej, Sayali Kulkarni, and Ashish V. Tendulkar. "Comparative Study of Clustering Techniques for Short Text Documents." In: *Proceedings of the 20th International Conference Companion on World Wide Web*. WWW '11. Hyderabad, India: ACM, 2011, pp. 111–112. ISBN: 978-1-4503-0637-9. DOI: 10.1145/1963192.1963249. URL: http://doi.acm.org/10.1145/1963192.1963249.

[6] L. AlSumait, D. Barbará, and C. Domeniconi. "On-line LDA: Adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking." In: *2008 Eighth IEEE International Conference on Data Mining*. Dec. 2008, pp. 3–12. DOI: 10.1109/ICDM.2008.140.

[7] Slobodan Beliga. "Keyword extraction: a review of methods and approaches." In: *University of Rijeka, Department of Informatics, Rijeka* (2014).

[8] Rada Mihalcea and Paul Tarau. "TextRank: Bringing Order into Text." In: *EMNLP*. Vol. 4. 2004, pp. 404–411.

[9] S LAHIRI, R MIHALCEA, and P.-H LAI. "Keyword extraction from emails." In: (Sept. 2016), pp. 1–23.

[10] M. F. Porter. "Readings in Information Retrieval." In: ed. by Karen Sparck Jones and Peter Willett. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997. Chap. An Algorithm for Suffix Stripping, pp. 313–316. ISBN: 1-55860-454-5. URL: http://dl.acm.org/citation.cfm?id=275537.275705.

[11] Anette Hulth. "Improved Automatic Keyword Extraction Given More Linguistic Knowledge." In: *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. EMNLP '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 216–223. DOI: 10.3115/1119355.1119383. URL: https://doi.org/10.3115/1119355.1119383.

[12] Sifatullah Siddiqi and Aditi Sharan. "Article: Keyword and Keyphrase Extraction Techniques: A Literature Review." In: *International Journal of Computer Applications* 109.2 (Jan. 2015). Full text available, pp. 18–23.

[13] Corina Florescu and Cornelia Caragea. "PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2017, pp. 1105–1115.

[14] Andras Csomai and Rada Mihalcea. "Investigations in Unsupervised Back-of-the-Book Indexing." In: (Jan. 2007), pp. 211–216.

[15] Charu C. Aggarwal and ChengXiang Zhai. "A Survey of Text Classification Algorithms." In: *Mining Text Data*. Ed. by Charu C. Aggarwal and ChengXiang Zhai. Boston, MA: Springer US, 2012, pp. 163–222. ISBN: 978-1-4614-3223-4. DOI: 10.1007/978-1-4614-3223-4_6. URL: https://doi.org/10.1007/978-1-4614-3223-4_6.

[16] Karen Sparck Jones. "A statistical interpretation of term specificity and its application in retrieval." In: *Journal of documentation* 28.1 (1972), pp. 11–21.

[17] Juan Ramos et al. "Using tf-idf to determine word relevance in document queries." In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. 2003, pp. 133–142.

[18] Emanuele Pianta and Sara Tonelli. "KX: A flexible system for keyphrase extraction." In: *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics. 2010, pp. 170–173.

[19] Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. "KEA: Practical automatic keyphrase extraction." In: *Proceedings of the fourth ACM conference on Digital libraries*. ACM. 1999, pp. 254–255.

[20] Taeho Jo, Malrey Lee, and Thomas M Gatton. "Keyword extraction from documents using a neural network model." In: *Hybrid Information Technology, 2006. ICHIT'06. International Conference on*. Vol. 2. IEEE. 2006, pp. 194–197.

[21] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, Nov. 1999. URL: http://ilpubs.stanford.edu:8090/422/.

[22] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. "Automatic keyword extraction from individual documents." In: *Text Mining: Applications and Theory* (2010), pp. 1–20.

[23] Zhen Yang, Jianjun Lei, Kefeng Fan, and Yingxu Lai. "Keyword extraction by entropy difference between the intrinsic and extrinsic mode." In: *Physica A: Statistical Mechanics and its Applications* 392.19 (2013), pp. 4523–4531.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed Representations of Words and Phrases and their Compositionality." In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger. Curran Associates, Inc., 2013, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

[25] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. "From Word Embeddings To Document Distances." In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 957–966. URL: http://proceedings.mlr.press/v37/kusnerb15.html.

[26] Seongwook Youn and Dennis McLeod. "A Comparative Study for Email Classification." In: *Advances and Innovations in Systems, Computing Sciences and Software Engineering*. Aug. 2007, pp. 387–391.

[27] Maria-Luiza Antonie, Osmar R. Zaïane, and Alexandru Coman. "Application of Data Mining Techniques for Medical Image Classification." In: *Proceedings of the Second International Conference on Multimedia Data Mining*. MDMKDD'01. London, UK, UK: Springer-Verlag, 2001, pp. 94–101. URL: http://dl.acm.org/citation.cfm?id=3012377.3012388.

[28] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs Up?: Sentiment Classification Using Machine Learning Techniques." In: *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 79–86. DOI: 10.3115/1118693.1118704. URL: https://doi.org/10.3115/1118693.1118704.

[29] Duyu Tang, Bing Qin, and Ting Liu. "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification." In: (Jan. 2015), pp. 1422–1432.

[30] Xiaojun Wan and Jianguo Xiao. "Single Document Keyphrase Extraction Using Neighborhood Knowledge." In: *AAAI*. Vol. 8. 2008, pp. 855–860.

[31] John A. Hartigan. *Clustering Algorithms*. 99th. New York, NY, USA: John Wiley & Sons, Inc., 1975. ISBN: 047135645X.

[32] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231. URL: http://dl.acm.org/citation.cfm?id=3001460.3001507.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of April 3, 2018 (`classicthesis`).