Keyphrase Extraction and Geospatial Characterizations for the Usage of Keyphrases

Francisco José Guerra Carreira

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Bruno Emanuel da Graça Martins

Prof. Miguel Daiyen Carvalho Won

October 2017

Acknowledgements

First, I would like extend my gratitude to Professor Bruno Emanuel da Graça Martins and Dr. Miguel Won for their significant contributions for this project, both in knowledge, and in guidance.

Second, I would like to thank my family for their unwavering support throughout the journey that concludes with this thesis. Above all else, they brought me here.

Third, I would like to thank Dr. Kenneth Heafield, the creator of the KenLM Language Modelling Toolkit, for guidance in adapting it for part of this project.

Fourth, I would like to thank all my friends, many of whom walked the same path, as well as the others who walked alongside it.

Finally, I would like to thank my colleagues under the tutelage of Professor Bruno Martins, for the company during the long working hours, the immeasurable number of trips to the coffee machine, and the maddening amount of laughter. They embodied the notion that help will always be given at N5.25 to those who ask for it.

Francisco José Guerra Carreira

And this, too, shall pass away. How much it expresses! How chastening in the hour of pride! How consoling in the depths of affliction!

Abraham Lincoln

Resumo

A extracção de palavras chaves é uma tarefa importante no processamento de lingua natural. Palavras chaves podem, por exemplo, facilitar o processo de resumir uma colecção de documentos ao descreverem cada documento com concisão. Adicionalmente, estas facilitam o processo de visualização de padrões que existem em documentos textuais, além de facilitar outras tarefas, como categorização, agrupamento, indexação e pesquisa. Existem duas categorias principais de métodos de extracção automática de palavras chave: a abordagem supervisionada que depende de dados de treino, e o método não supervisionado que tenta inferir a relevância de palavras chave a partir de estatísticas directamente extraídas de um conjunto de dados. Este projecto avança uma proposta para um novo método não supervisionado para a extracção de palavras chave, combinando uma abordagem baseada em medidas de centralidade sobre um grafo ponderado com técnicas de modelos linguísticos. O novo método avança uma combinação inovadora de diferentes abordagens para estimar a importância de uma palavra chave e o grau da relação semântica entre candidatos a palavras chave. O novo método incorporou ainda técnicas utilizadas para estimar o grau de autocorrelação espacial no uso de palavras chave, tendo como objectivo a captura da palavras chave candidata com padrões de distribuição espacial interessantes (por exemplo, palavras chaves específicas de uma dada região). Testamos os novos métodos em três corpora diferentes frequentemente usados para avaliar os métodos de extracção de palavras chave. Os resultados da avaliação indicam que os resultados obtidos com o novo método aproximam e, em alguns casos, superam os resultados obtidos por outros métodos estado-da-arte para extracção de palavras chave.

Abstract

Keyphrase extraction is an important task in natural language processing. Keyphrases can, for instance, ease the process of summarizing a collection of documents by concisely describing each document. Furthermore, they facilitate the process of visualizing patterns that exist in textual documents, in addition to facilitating other tasks such as, categorization, clustering, indexing and searching. There are two main categories of automatic keyphrase extraction methods: the supervised approach that relies on training data, and the unsupervised method which tries to infer keyphrase relevance from statistics directly collected from a dataset. This project advances a proposal for a novel unsupervised method for keyphrase extraction, combining an approach based on centrality over a weighted graph with language modeling techniques. The new method will thus leverage an innovative combination of different approaches for estimating the importance of a keyphrase and the strength of the semantic relation between candidate keyphrases. Techniques used to estimate the degree of spatial auto-correlation in the usage of keyphrases were also incorporated into the new method, the goal being the capture of candidate keyphrase with interesting spatial distribution patterns (e.g., that are specific to a particular region). We tested the new methods on three different corpora commonly used for evaluating keyphrase extraction methods. The evaluation results indicate that this technique can approximate (and in some cases surpass) the results obtained by other state-of-the-art keyphrase extraction methods.

Palavras Chave
Keywords

Palavras Chave

Extracção de palavras chaves

Medidas de centralidade em grafos

Modelos linguísticos

I de Moran

Heurísticas para extracção de palavras chaves

Embeddings de palavras

Keywords

Keyphrase extraction

Graph centrality measures

Language models

Moran’s I

Heuristics for keyphrase extraction

Word embeddings

Contents

List of Figures

List of Tables

1 Introduction

Keyphrase extraction is the process of extracting important and highly descriptive phrases (i.e., sequences of words such as named entities) from a textual document. Keyphrases can, for instance, facilitate the process of summarizing the contents of a collection of documents by concisely describing each document. The use of keyphrases can also ease the process of visualizing and analyzing patterns that exist in textual information but are usually dispersed throughout several documents. Furthermore, keyphrases are useful for document categorization, clustering, indexing, and searching.

Two broad categories of methods are primarily used for automatic keyphrase extraction: the supervised approach that relies on training data to infer a function that maps a set of features (i.e., input values) to some known output; and unsupervised learning methods which try to infer the structure of a dataset without the use of training data. There are advantages to using unsupervised methods. First, they do not require that the dataset be annotated for training, thus increasing the range of application to include large collections of unprocessed texts and lowering the human cost involved in the development of the method. Second, since they do not require training the model, they tend to be as computationally cheaper. However, despite their usefulness, automatic keyphrase extraction systems still have a poor performance when compared to systems that address other Natural Language Processing (NLP) tasks (Hasan and Ng, 2014).

This Chapter provides an introduction to my M.Sc. project, being divided into the following sections: Section 1.1 provides an overview of the objectives of this project, as well as a summary of the method that was developed; Section 1.2 describes the work methodology used in the development of this project; Section 1.3 summarizes the datasets used for testing the developed method, the results obtained on those datasets, and the contributions that were made in the development of this project; Section 1.4 details the outlining structure of this document.

1.1 Objectives

Recent studies on NLP have proposed methods for unsupervised keyphrase extraction that combine different heuristics to model the relations between candidate terms. My M.Sc. project addressed the development of a novel unsupervised method for keyphrase extraction, combining an approach based on centrality over a weighted graph, with language modeling techniques. The new method leverages an innovative combination of different approaches for estimating the importance of a keyphrase in a given document, and the strength of the semantic relation between candidate keyphrase (i.e., the weight of the connections between candidate keyphrases in a graph). Results indicate that the new method can approximate and, in some cases, surpass, comparable state-of-the art keyphrase extraction methods. Additionally, the developed method also integrates techniques to estimate the degree of spatial auto-correlation in the usage of keyphrases. This latter aspect is particularly innovative, and the idea is to capture candidate keyphrases whose spatial distribution follows particular patterns (e.g., keyphrases that are specific to a particular region). The final implementation is able to process textual documents that contain geographic annotations, identify keyphrases and perform a geographic characterization of the candidates.

To summarize, the goal of this thesis was the development of this new keyphrase extraction method, to test several combinations of the heuristics used in this method, to assess which ones are effective at increasing its performance, and to study how this method compares with other state-of-the art keyphrase extraction methods on several corpora with different characteristics.

1.2 Methodology

The first stage on the development of this project consisted of collecting multiple datasets which would be used on a later stage to evaluate the performance of the new method. These datasets needed to be analyzed for completeness (i.e., are all documents present), format (i.e., how are the texts/annotation arranged), and encoding.

In the second stage, the related work was studied to see which heuristics could be interesting to include in this project. Following the study of the related word, I studied which software packages were available for inclusion, which of those packages would be suitable for this project (for example, which parts-of-speech (POS) tagger, stemmer, and lemmatizer should be used), what would need to be programmed, and which programming language should be used. The main programming language chosen for this project was Python due the wide availability of software packages relating to NLP, the large number of statistical tools, and the ease of development. In addition to Python, some other languages were used to address specific needs, these were the following: Java, C++, and R.

Following the development of the code that implemented the method proposed in this project, in the third stage, testing was done on several datasets, the performance of the new keyphrase extraction method was measured, and parameters were adjusted to improve its performance.

The fourth stage consisted on the realization of a case study, in which, a spatial autocorrelation measure, the Moran's I statistic (Moran, 1950), was incorporated into the new method, and testing was done on a corpus annotated with spatial information about the texts it contained. This corpus was created at this stage from a publicly available dataset containing legends and urban myths collected in Portugal.

In the fifth and final stage, conclusions were drawn about the viability of the new method, how it compares to other state-of-the art keyphrase extraction methods, and what future work could be interesting for this project.

1.3 Results and Contributions

The main contributions of this thesis are as follows:

- A new unsupervised keyphrase extraction method that combines graph centrality methods, with multiple heuristic for estimating a prior probability for each candidate, as well as a novel approach for estimating the semantic relation between candidates in the graph (i.e., the weight of the edges in the graph).
- An analysis of the evaluation that was performed of the new method, on three corpus annotated with gold standard keyphrases: the SemEval-2010 dataset Kim et al. (2010) which a corpus containing 244 conference and workshop papers, the Inspec dataset Hulth (2003) which contains 2000 abstracts, and the DUC2001 dataset (Over and Yen, 2001) that consists of 308 news articles.
- A comparison between the performance of the new method and other state-of-the art related works.
- A case study in which a spatial autocorrelation metric was combined with the new method as a way to extract candidate keyphrases with a geographic uniqueness to them, and

a discussion on how spatial autocorrelation metrics could benefit keyphrase extraction methods.

1.4 Outline of the Document

This paper is organized in the following way:

- Chapter 2 provides a summary of previous research related to unsupervised keyphrase extraction.
- Chapter 3 details the proposed method.
- Chapter 4 details how the proposed method was evaluated, on which corpora, and how it compares to other state-of-the art methods.
- Chapter 5 presents the conclusion and ideas for future work.

2 Concepts and Related Work

This chapter describes fundamental concepts and previous studies addressing the taks of keyphrase extraction. Section 2.1 presents some fundamental concepts used by several keyphrase extraction methods. Section 2.2 describes previous works in keyphrase extraction. Finally, a summary is presented at the end of this chapter.

2.1 Fundamental Concepts

Keyphrase extractions methods rely on fundamental tasks in natural language processing for extracting candidates from a text, e.g. techniques to label a sequence of symbols with some useful label. This section provides a description of some of those fundamental tasks, and the algorithms commonly used to address them.

2.1.1 Sequence Tagging for Natural Language Processing

Sequence tagging is the process of assigning a label (i.e., a class) to each symbol in a sequence (e.g., a sequence of word tokens). This section describes structured learning methods commonly used in solving the task of sequence tagging.

Hidden Markov Models (HMMs) are a statistical tool for representing probability distributions over a generative sequence (Ghahramani, 2001). The model derives its name from two properties; first, we assume that an observation (e.g. a word) made at time t was generated by a process whose state i_t is hidden from the observer, representing an unknown class; second, we assume that the value of state i_t depends only on i_{t-1} (i.e., Markov property). The outputs also satisfy the Markov property in that the output symbol O_t depends only on the state i_t. If we let I be defined as the sequence of states $(i_1, i_2, ..., i_t)$, and O the sequence of observed symbols $(O_1, O_2, ..., O_t)$, the joint distribution of a sequence of states and observations is given as:

$$\mathrm{P}(I, O) = \mathrm{P}(I_1) \times \mathrm{P}(O_1|I_1) \prod_{t=2}^{T} \mathrm{P}(I_t|I_{t-1}) \times \mathrm{P}(O_t|I_t) \tag{2.1}$$

To use this model we first need to define some additional notation. Let N be the number of states in the model, M the number of distinct observation symbols, T the length of the observation sequence, and V the set of observation symbols $\{v_1, ..., v_n\}$. Let also $\pi_i = \text{P}(i_1 = i)$, with $\pi = \{\pi_i\}$, be the probability of being at state i at the beginning of the experiment, Finally $A = \{a_{ij}\}$ is the transition probability matrix where $\{a_{ij}\} = \text{P}(i_{t+1} = j | i_t = i)$ (i.e., the probability of being in state j at time t given that we were in state i at time $t-1$) and $B = \{b_j(k)\}$ is the probability of observing symbol v_k given that we are in state j, i.e., P(v_k at $t | i_t = j$). A HMM can be seen as a tuple $\lambda = (A, B, \pi)$. For sequence tagging, the problem can now be defined as leveraging the HMM for finding the most probable sequence of states I that generates the observed sequence of symbols O. This is known as the decoding problem:

$$\arg\max_I \text{P}(O, I|\lambda) \tag{2.2}$$

The direct approach to solving Equation 2.2 is to enumerate all possible sequences for I and solve the probability function. However, this method of solving the problem is computationally very expensive. We nonetheless know that:

$$\begin{aligned} \text{P}(O, I|\lambda) &= \text{P}(\lambda|O, I)\text{P}(\lambda) \\ &= \pi_{i_1} b_{i_1}(O_1) a_{i_1 i_2} b_{i_2}(O_2) ... a_{i_{T-1} i_T} b_{i_T}(O_T) \end{aligned} \tag{2.3}$$

In order to avoid problems of numerical precision, associated to long sequences of multiplications of probabilities, the previous expression is often re-written trough the use of the logarithm function:

$$U(i_1, i_2, ..., i_T) = -[\ln(\pi_{i_1} b_{i_1}(O_1)) + \sum_{t=2}^{T} \ln(a_{i_{T-1} i_T} b_{i_T}(O_T))] \tag{2.4}$$

$$\text{P}(O, I|\lambda) = \exp(-U(i_1, i_2, ..., i_T)) \tag{2.5}$$

Based on Equations 2.3, 2.4 and 2.5, the problem now becomes solving the following equation:

$$\arg\min_{\{i_t\}_{t=1}^{T}} U(i_1, i_2, ..., i_T) \tag{2.6}$$

We can solve this problem using the Viterbi algorithm (Nguyen and Guo, 2007) which is a dynamic programming approach to computing least cost paths.

Leaning the parameters of the model can be done in various ways. The following description explains how the parameters can be estimated in a supervised setting using training data. We need to estimate the transition probabilities $\{a_{i_j}\}$ by counting the number of transitions from a state i to a state j, and dividing by the total number of transitions in the training data (Equation 2.7).The emissions probabilities $b_j(k)$ can be calculated by dividing the number of times a state j emitted the symbol o_k, by the number of emissions of state j (Equation 2.8). The beginning probabilities π_i are computed in the same way as the transition probabilities, but counting the number of times there is a transitions from the start (denoted by the symbol $\oslash$) to a state i (Equation 2.9).

$$\hat{\mathrm{P}}(i \to j) = \frac{\mathrm{c}(i \to j) + 1}{\sum_{s \in I} \mathrm{c}(i \to s) + |I|} \tag{2.7}$$

$$\hat{\mathrm{P}}(j \uparrow o_k) = \frac{\mathrm{c}(j \uparrow o_k) + 1}{\sum_{\rho \in O} \mathrm{c}(j \uparrow \rho) + |\oslash|} \tag{2.8}$$

$$\hat{\mathrm{P}}(\oslash \to i) = \frac{\mathrm{c}(\oslash \to i) + 1}{\sum_{s \in I} \mathrm{c}(\oslash \to s) + |I|} \tag{2.9}$$

In Equations 2.7 and 2.8, $\mathrm{c}(i \to j)$ is the number of times a transition from state i to state j occurred, and $\mathrm{c}(j \uparrow o_k)$ is the number of times state j emitted the symbol o_k. Laplace smoothing was applied to all counts to prevent division by zero. As such, all unseen symbols have the same probability estimate.

The Viterbi is a dynamic programming algorithm used to compute least cost paths. Reusing the notation defined for the HMMs, and let S be the state space $S = (s_1, ..., s_k)$, the pseudocode for the Viterbi algorithm is defined as follows:

Algorithm 1: The Viterbi algorithm pseudocode.

```
function VITERBI( $V$ , $S$ , $\pi$ , $O$ , $A$ , $B$ ) : $I$
    foreach state $s_i$ do
        T1[i,1] $\leftarrow \pi_i \cdot B_{iO_1}$
        T2[i,1] $\leftarrow 0$
    end for
    for i $\leftarrow$ 2,3,...,T do
        foreach state $s_j$ do
            $T_1[j,i] \leftarrow \max_k (T_1[k,i-1] \cdot A_{kj})$
            $T_1[j,i] \leftarrow (T_1[j,i] \cdot B_{jO_i})$
            $T_2[j,i] \leftarrow \arg\max_k (T_1[k,i-1] \cdot A_{kj})$
        end for
    end for
    $z_T \leftarrow \arg\max_k (T_1[k,T])$
    $x_T \leftarrow s_{z_T}$
    for $i \leftarrow T, T-1, ..., 2$ do
```

```
            $z_{i-1} \leftarrow T_2[z_i, i]$
            $I_{i-1} \leftarrow s_{z_{i-1}}$
        end for
        return $I$
    end function
```

The cost reduction using the Viterbi algorithm shown in Algorithm 1 comes from the 2-dimensional $K \times T$ matrices T_1 and T_2. Each element $T_1[i,j]$ store the probability of the most likely path so far $\hat{I}$ =$(\hat{i}_1, ..., \hat{i}_j)$, with $\hat{i}_j = s_i$ that generates $O = (O_1, ...O_l)$. Each element $T_2[i,j]$ stores $\hat{I}_{j-1}$ of the most likely path so far $\hat{I} = (\hat{i}_1, ..., \hat{i}_{j-1})$ for $\forall j, 2 \leq j \leq T$

Structured perceptrons are an extension to the standard perceptron proposed by Collins (2002) that can also be used to perform sequence tagging. Having as input a sequence of symbols $O = (O_1, O_2, ..., O_t)$, and a sequence of corresponding labels $I = (i_1, i_2, ..., i_t)$, we first generate a set of label sequences $\mathcal{Y}$ which represents all possible sequences of labels for the input sequence O. We can then compute the prediction score as follows:

$$\hat{I_n} = \arg\max_{I\prime \in \mathcal{Y}} \phi(O, I\prime) \cdot \mathbf{w} \tag{2.10}$$

In Equation 2.10, ϕ is a function from $(O_{[1:t]}, i_{[1:t]})$ to a d-dimensional feature vector, and $\mathbf{w}$ is a weight vector of size d. The arg max function can be solved using the Viterbi algorithm. When training the model, if $\hat{I} \neq I_n$ (i.e., if the predicted label sequence does not match the correct labels in the training data) then we adjust the weight vector $\mathbf{w}$ as follows:

$$\mathbf{w} = \mathbf{w} + \phi(O, I) - \phi(O, \hat{I}) \tag{2.11}$$

To train the model we apply Equation 2.10 and Equation 2.11 to all the input sequences in the training data.

Conditional Random Fields (CRFs) are undirected graphical models that given an observation O and random variables I encoding class labels, are trained to maximize $\mathrm{P}(I|O)$. A linear-chain CRF is a special case used for sequence labeling (Nguyen and Guo, 2007; Sutton and McCallum, 2012), where O and I are both sequences. Let $\mathbf{w}$ be the parameters of a linear-chain CRF. The probability of a sequence of states $I = i_1, i_2, ..., i_t$ given $O = O_1, O_2, ..., O_t$ as input can be computed as shown in Equation (2.12):

$$\mathrm{P}(I|O) = \frac{1}{Z_{\mathbf{w}}} \exp(\mathbf{w} \cdot \phi(O, I\prime)), \tag{2.12}$$

In Equation 2.12, $Z_{\mathbf{w}}$ is a normalization constant and the parameters are estimated by maximizing Equation 2.13, which represents the log-likelihood of the training set penalized by a Gaussian prior over the parameters, and with σ_d^2 being the variance of the Gaussian distribution associated to a parameter d (Nguyen and Guo, 2007).

$$\sum_i \log\mathrm{P}(\boldsymbol{y}_i|\boldsymbol{x}_i) - \sum_d \frac{w_d^2}{2\sigma_d^2} \tag{2.13}$$

2.1.2 Part-of-Speech Tagging

Part-of-Speech (POS) tagging is the process of assigning morpho-syntactic tags to words in a text. Tags provide useful information about the morphology of a word and the syntactical structure of the sentence that contains it. Examples of POS tags are lexical categories like verbs (VBP), prepositions (PRP) or nouns (NN) (Jurafsky and Martin, 2000). Supervised POS tagging methods rely on a corpus annotated with POS tags, defined in a given tagset, from which we can compute the probability that a word should be assigned a given tag. One such corpus is the Brown Corpus (Mitchell P. Marcus and Santorini, 1993), containing English text with more than a million annotated words. Multiple POS tagsets exist for different languages and a universal tagset was proposed by Petrov et al. (2012), consisting of twelve universal POS categories common to 22 languages. Three main approaches are used in POS tagging: rule-based approaches use hand-crafted rules to tag a text, assigning all possible tags to a word with the help of a dictionary, and then using disambiguation rules to choose the correct label; transformation based taggers are based on a hybrid approach that assigns tags according to probabilities extracted from the corpus, and then uses rules to resolve any ambiguity and correct mislabeled words (Brill, 1995); the stochastic approach aims to correctly tag a sequence of words by leveraging sequential models such as those introduced in Section 2.1.1, which have their parameters inferred from training data.

2.1.3 Noun Phrase Chunking

Noun Phrase (NP) chunking is the process of extracting, from a given text, chunks that correspond to individual noun phrases. A chunk is a grouping of multi-token sequences that do not overlap in the text. For example, the sentence *we saw the yellow dog* contains two noun phrases, namely, *we* and *the yellow dog* (Steven Bird and Loper, 2009). Noun phrase chunking is one of the motivations for POS tagging in the context of information extraction systems, given that a tagged text can be processed by grammar rules that define how sentences should be chunked. Using this approach we can, for instance, create a chunk parser with the following rule: determiner (DT) followed by adjectives (JJ) and a noun. A regular expression that represents the previous example would be `NP: {<DT >?<JJ >*< NN>}`. Parsing a sentence *the* (DT) *little* (JJ) *yellow* (JJ) *dog* (NN) *barked* (VDB) *at* (IN) *the* (DT) *cat* (NN), would yield as noun phrases the sub-sentences *the little yellow dog* and *the cat.*

NP chunking can also be modeled as a task of extracting non-overlapping sequences of symbols from a text through a previously trained statistical model. In this case, we need a corpus in which words are annotated with labels that indicate their position in relation to a noun phrase (i.e., indicate if a word is inside, outside, or at the beginning of a NP). A corpus annotated with these types of labels, commonly referred to as IOB tags, allows us to leverage the techniques described in Section 2.1.1, in which the goal is to assign IOB tags to the words in a target text. NPs are generated by combining non-overlapping sequences of consecutive words, in which the first word of the sequence is labeled B (i.e., a tag indicating the beginning of a NP), and the following words have the label I (i.e., a tag indicating the inside of a NP).

2.2 Related Work on Unsupervised Keyphrase Extraction

This section contains a description of existing work on unsupervised keyphrase extraction. Section 2.2.1 provides an explanations of graph-based techniques for keyphrase extraction, and some implementations that leverage those techniques, such as TextRank or SGRank. Section 2.2.2 contains a description of language modeling techniques for keyphrase extraction, particularly the method proposed by Tomokiyo and Hurst (2004), for estimating the phraseness and informativeness of candidate keyphrases, in addition to smoothing methods for improving the estimation of n-gram probabilities.

2.2.1 Graph-Based Techniques for Keyphrase Extraction

Graph-based keyphrase extraction algorithms create a graph representation of a text in order to extract potential keyphrases. Usually, a graph is created in which candidate keyphrases for a text are represented as nodes and the co-occurrences between them are represented as edges. The nodes are then ranked using a graph centrality measure that captures the semantic importance of candidates in the text (Danesh et al., 2015).

In the graph-based approach described by Mihalcea and Tarau (2004), the target text is first annotated using POS (part-of-speech) tags and passed through a syntactic filter. A filter can consider only nouns, nouns and verbs, or any combination of tags with useful semantic links. The remaining lexical units are added as nodes to the graph (i.e., this particular method builds a graph where nodes correspond to words) and an edge is added between nodes if the lexical units they represent co-occur in the text within a given window of N words.

After generating the graph, a centrality measure is used to assign a weight to each node. The top T candidates are extracted for post processing, during which, the top ranked candidates are marked in the original text as a way to collapse sequences of adjacent candidates into multi-word keyphrases. It is worth noting that Mihalcea and Tarau (2004) propose to set T to one-third of the number of nodes in the graph. The method by Mihalcea and Tarau (2004) uses PageRank as the centrality measure, but different measures can be used to estimate how central each node is. Examples include:

Degree centrality: The weight of each node is calculated based on the number of edges that are connected to it. For a node V_i from the set of nodes V, with $|\mathrm{E}(V_i)|$ edges, its degree centrality is computed as follows:

$$\mathrm{C_D}(V_i) = \frac{|\mathrm{E}(V_i)|}{|V|} \tag{2.14}$$

Closeness centrality: Node weights represent how close each node is to every other node in the graph, and these weights can be computed from the sum of the length of all the closest paths from the target node V_j to all other nodes V_i. With $\mathrm{Dist}(V_i, V_j)$ being the shortest distance between two nodes, this measure is computed as follows:

$$\mathrm{C_C}(V_i) = \frac{|V| - 1}{\sum_{V_j \in V} \mathrm{Dist}(V_i, V_j)} \tag{2.15}$$

Betweenness centrality: We can also compute how many shortest paths pass through the node V_i, and use this quantity as a measure of centrality. Let $\sigma(V_j, V_k)$ be the number of shortest paths between nodes V_j and V_k, and let $\sigma(V_j, V_k|V_i)$ denote those paths that pass through node V_i. With this, we can compute the betweenness centrality as follows:

$$\mathrm{C_B}(V_i) = \frac{\displaystyle\sum_{V_i \neq V_j \neq V_k \in V} \frac{\sigma(V_j, V_k|V_i)}{\sigma(V_j, V_k)}}{(|V| - 1)(|V - 2|)/2} \tag{2.16}$$

Eigenvector centrality: More advanced procedures take into account not only the number of edges that connect to a node, but also the score of each node connected to the target, leveraging the assumption that nodes which are scored higher should contribute more. The score of each node is initialized with a fixed value and is re-calculated recursively until convergence is achieved. With W_{ij} as the weight of the edge between V_i and V_j, and with λ as a constant, this notion of centrality can be computed through the following recursive definition:

$$\mathrm{C_E}(V_i) = \frac{1}{\lambda} \sum_{V_j \in \mathrm{E}(V_i)} W_{ij} \times \mathrm{C_E}(V_j) \tag{2.17}$$

PageRank centrality: The PageRank metric, originally proposed by Page et al. (1999) is very similar to eigenvector centrality. It incorporates a damping factor d, usually set to 0.85 (Page et al., 1999), that integrates the probability of randomly jumping from a given node to another random node in the graph. Formally, the score $\mathrm{C_P}$ of a of a node V_i, having $\mathrm{In}(V_i)$ as the set of nodes that point to V_i, can be computed as follows:

$$\mathrm{C_P}(V_i) = \frac{(1-d)}{|V|} + d \times \sum_{V_j \in \mathrm{In}(V_i)} \frac{1}{|\mathrm{Out}(V_j)|} \times \mathrm{C_P}(V_j) \tag{2.18}$$

Although applied traditionally to directed graphs, the recursive formula in Equation 2.18 can also be used for undirected graphs, in which case, the out-degree of a node is equal to its in-degree.

Weighted PageRank centrality: Adapted versions of the PageRank method can allow the incorporation of weighted edges representing the strength of a connection between two nodes, and/or weighted nodes. Having w_{ij} as the weight of the edge between V_i and V_j, the weighted score $\mathrm{C_{WP}}$ of a node can for instance be computed as follows:

$$\mathrm{C_{WP}}(V_i) = \frac{(1-d)}{|V|} + d \times \sum_{V_j \in \mathrm{In}(V_i)} \frac{w_{ji}}{\sum_{V_k \in \mathrm{Out}(V_j)} w_{jk}} \times \mathrm{C_{WP}}(V_j) \tag{2.19}$$

There are many studies, besides TextRank that have attempted to address the problem of keyphrase extraction by leveraging graph-based centrality measures. The following paragraphs describe some of the existing methods that use this approach.

SGRank (Danesh et al., 2015) is a keyphrase extraction algorithm that combines graph-based techniques with statistical heuristics in order to rank candidate keyphrases. The algorithm processes an input document in four stages. First, all n-grams up to length 6 are extracted from the text. From this list, n-grams are removed if (i) they are not sequences of verbs, nouns or adjectives, if (ii) they contain punctuation marks or stop-words, or if (iii) they have a frequency bellow a given threshold. In the second stage, candidates are ranked using a modified version of

TF-IDF, as used in the KP-Miner system (El-Beltagy and Rafea, 2009), in which n-grams longer than 1 are considered to have a document frequency of 1 in IDF calculations (i.e., multi-word candidates are considered to occur in a single document). According to Danesh et al. (2015), the intuition behind the modified TF-IDF is that rareness, as an indication of semantic importance, is more applicable in the case of single words than in multi-word expressions. In the third stage, the top ranked T candidates are re-ranked based on additional heuristics. These heuristics are the position of first occurrence, term length, and subsumption count. For the position of first occurrence factor (PFO), the authors propose a novel encoding using a logarithmic decay function computed as follows:

$$\mathrm{PFO}(t,d) = \log\left(\frac{cp}{\mathrm{p}(t,d)}\right) \tag{2.20}$$

In Equation 2.20, cp is a cutoff position set to 3000, and $\mathrm{p}(t,d)$ is the position of the term t inside a document d. Knowing that a term t is subsumed by a term t_s if t contains t_s, we can now compute the weight $\mathrm{W_s}$ for a term t as follows:

$$\mathrm{W_s}(t,d) = (\mathrm{TF}(t,d) - \mathrm{SC}(t,d)) \times \mathrm{IDF}(t) \times \mathrm{PFO}(t,d) \times \mathrm{TL}(t) \tag{2.21}$$

In Equation 2.21, $\mathrm{TL}(t)$ is the term length, and $\mathrm{SC}(t,d)$, is the subsumption count for term t (i.e., the sum of term frequencies of all terms in the top ranked list that subsume the term t).

Finally, in the fourth stage, a graph ranking approach that leverages the weighted PageRank centrality measure is used. Candidates with positive weight after stage three are added to a graph, and an edge is added between candidates if they co-occur within a window of 1500 words. The weight $\mathrm{W_d}$ of the distance between two nodes t_i and t_j, is attenuated based on an average log decayed distance between co-occurrences of the terms pair, and is computed as follows:

$$\mathrm{W_d}(t_i,t_j) = \frac{\sum_{i=1}^{\mathrm{TF}(t_i)} \sum_{i=1}^{\mathrm{TF}(t_j)} \log\left(\frac{ws}{|pos_i - pos_j|}\right)}{\mathrm{NCO}(t_i,t_j)} \tag{2.22}$$

In Equation 2.22, $\mathrm{NCO}(t_i,t_j)$ is the number of co-occurrences between candidates t_i and t_j within a window of 1500 words, while pos_i and pos_j are the positions of occurrence of the candidates t_i and t_j, respectively. Furthermore, the statistical weights computed in Equation 2.21 are included in the final edge weights $\mathrm{W_e}$, which are computed as follows:

$$\mathrm{W_e}(t_i,t_j) = \mathrm{W_d}(t_i,t_j) \times \mathrm{W_s}(t_i) \times \mathrm{W_s}(t_j) \tag{2.23}$$

Having w_{ji} as the weight $\mathrm{W_e}(t_j,t_i)$, and w_{jk} as the weight $\mathrm{W_e}(t_j,t_k)$, the final score $\mathrm{C_{WP}}$ for each candidate node V_i can now be computed using Equation 2.19. Danesh et al. (2015) claim that SGRank can significantly outperform TextRank, and give as comparison the F1-score obtained on a SemEval dataset (Kim et al., 2010): they achieved an F1-score of 27.20 for K=15, versus an F1-score of 3.47 for TextRank.

Wan and Xiao (2008) proposed a keyphrase extraction method where the score of a candidate depends on two main factors, the centrality measure that candidate has on a graph representation of the target document (i.e., the document from which we are extracting keyphrases), and the centrality measure on the graph representation of a neighbourhood of documents. They start by computing the similarity of all documents in a corpus $D = \{d_1, d_2, ..., d_n\}$. They compute the pairwise similarity between documents d_i and d_j as the cosine similarity of the term vectors $\mathbf{d_i}$ and $\mathbf{d_j}$ of each document:

$$\mathrm{sim}(d_i, d_j) = \frac{\mathbf{d_i} \cdot \mathbf{d_j}}{||\mathbf{d_i}|| \cdot ||\mathbf{d_j}||} \tag{2.24}$$

Following the computation of the similarity between all pairs of documents in the corpus, their method performs a neighborhood level evaluation of the candidate keyphrases as follows. For each set of neighbourhood documents $ND = \{d_0, d_1, d_2, ..., d_k\}$, where d_0 is the document from which we are extracting keyphrases, and d_1 through d_k are the k nearest neighbours to d_0, they extract all candidates from ND following the extraction process described by Mihalcea and Tarau (2004). Then, weighted edges are added between candidates that co-occur within a given window of N words in any of the documents in ND. The weight of these edges, namely, the affinity weight $\mathrm{Aff}(v_i, v_j)$, is simply set to be the count of the co-occurrences between the words v_i and v_j in the whole document set as follows:

$$\mathrm{Aff}(v_i, v_j) = \sum_{d_k \in ND} \mathrm{sim}(d_0, d_k) \times \mathrm{Count}_{d_k}(v_i, v_j) \tag{2.25}$$

In Equation 2.25, Count_{d_k} is the count of the controlled co-occurrences between words v_i and v_j in document d_k. The resulting graph, built over the whole document set, reflects the global information in the neighbourhood set. The similarity factor $\mathrm{sim}(d_0, d_k)$ is used to reflect the confidence value for using document d_k in the expanded set. The score $\mathrm{C_{WP}}(v_i)$ of each candidate is given by a PageRank centrality measure described in Equation 2.19, with d set to 0.85. Then, the candidate words (i.e., nouns and adjectives) of d_0, which is a subset of V, the set of all candidates in the neighbourhood graph, are marked in the text of the document d_0. Finally, for every phrase p_i starting with a noun or adjective, and ending in a noun, we compute the final score of p_i as the sum of the scores of its constituent words:

$$\text{Score}(p_i) = \sum_{v_j \in p_i} \text{C}_{\text{WP}}(v_i) \tag{2.26}$$

The algorithm then returns the top m phrases with the highest Score. The results obtained by this method are reported in Section 4.3, Table 4.3. The authors also propose a baseline for comparison, the SingleRank method, which is a variation of TextRank that weighs the edges with the co-occurrences between vertices, and no longer assembles keyphrases by collapsing adjacent candidates, instead, candidates are noun phrases extracted from the document and ranked according to the sum of the score of the words they contain.

TopicRank is an approach proposed by Bougouin et al. (2013) in which candidate keyphrases are clustered into topics. The first step consists of identifying topics that describe the text, by extracting noun phrases that identify them. As such, the authors extracting the longest sequences of nouns and adjectives from the document, and then group similar noun phrases as single entity (i.e., a topic). Two candidate keyphrase are considered similar if they have at least 25% overlapping words. To group similar candidates they use a Hierarchical Agglomerative Clustering (HAC) algorithm. Then, a graph is constructed where topics are represented as vertices, and edges are weighted according to the strength of the semantic relations between vertices. The semantic relation between two topics is considered strong if the candidate keyphrases they contain often appear together. Let $t_i = \{p_1, p_2, ..., p_i\}$ and $t_j = \{p_1, p_2, ..., p_j\}$, where t_i and t_j are topics containing the noun phrases p_i and p_j respectively, then the weight of the edge between topics is computed as follows:

$$\text{Weight}(t_i, t_j) = \sum_{p_i \in t_i} \sum_{p_j \in t_j} \text{Dist}(p_i, p_j) \tag{2.27}$$

$$\text{Dist}(\text{p}_\text{i}, \text{p}_\text{j}) = \sum_{po_i \in \text{pos}(p_i)} \sum_{po_j \in \text{pos}(p_j)} \frac{1}{|po_i - po_j|} \tag{2.28}$$

In Equation 2.28, $\text{Dist}(p_i, p_j)$ refers to the reciprocal distances between the offset positions of the candidate keyphrases p_i and p_j in the document, and $\text{pos}(p_i)$ represents all the offset positions of the candidate keyphrase p_i. If we let the t_i be equal to V_i, then we can compute the final score of each topic t_i using the weighted PageRank centrality measure defined in Equation 2.19. Finally, for each topic, they extract the most significant candidate keyphrase. The strategy for selecting the most significant candidate in a topic, which yielded the best results, was extracting from each topic the candidate that appears first in the text. The results obtained by this method are reported in Section 4.3, Table 4.3.

PositionRank is a biased variation TextRank proposed by Florescu and Caragea (2017). In their approach, candidate candidate keyphrases (i.e., nouns or adjectives) are given a prior weight based on their positions in a text from which keyphrases are being extracted. Two variations of the weight are presented by the authors. First, the prior weight only take into account the position of first occurrence po_1 of a given candidate p in the text. This first variation of the prior weight is computed as follows:

$$\text{Prior}(p) = \frac{1}{po_1} \tag{2.29}$$

In the second variation, the weight is computed as a weighted sum of all the positions in which the candidate appears in the text. Let $\mathbf{Pos} = \{po_1, po_2, ..., po_i\}$ be a set where po_i is the i^{th} position in which a candidate p appears in the text, then the prior weight of p is computed as follows:

$$\text{Prior}(p) = \sum_{po_i \in \mathbf{Pos}} \frac{1}{po_i} \tag{2.30}$$

Finally, the score of each candidate is computed using a weighted PageRank centrality measure, with a damping parameter d of 0.85, and the weight of the edges, Weight(p_i, p_j), is the number of co-occurrences between candidates p_i and p_j within a window of 2 words. To account for the prior weight of each candidate, the score is computed as follows:

$$\text{S}(p_i) = (1-d) \times \frac{\text{Prior}(p_i)}{\sum_{p_j} \text{Prior}(p_j)} + d \times \sum_{p_j \in \text{Links}(p_i)} \frac{\text{S}(p_j) \times \text{Weight}(p_i, p_j)}{\sum\limits_{p_k \in \text{Links}(p_j)} \text{Weight}(p_j, p_k)} \tag{2.31}$$

Florescu and Caragea (2017) report an F1 statistic for their method, for the top scoring 8 candidates, of 0.073, 0.106, and 0.121, in the KDD (Caragea et al., 2014), WWW (Caragea et al., 2014), and Nguyen (Nguyen and Guo, 2007) corpora, respectively.

Wang et al. (2014) proposed a weighing scheme that computes both phraseness (i.e., how much a sequence of words can be considered as phrases) and informativeness (i.e., how well the phrase captures key ideas) using information provided by word embedding vectors (Collobert et al., 2011) and local statistics. Word embeddings are typically low dimensional vector representations of features of a word. These features can encode both semantic and syntactic information (Wang et al., 2014). Mikolov et al. (2013) demonstrated that embedding vector calculations can capture semantic relations like *King-Man+Woman≈Queen.* Figure 2.1 illustrates

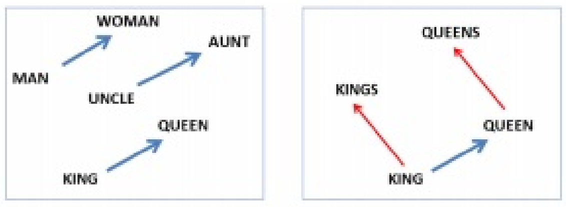

Figure 2.1: Left panel shows vector offsets illustrating gender relations between three word pairs. Right panel adds singular/plural relations to the representation of two word pairs. Both figures were taken from Mikolov et al. (2013).

these relationships. The similarity between two candidates can be computed as the euclidean distance between the average embedding vectors of the candidate words. They use word embeddings to measure the semantic relation between pairs of words and take into account the term frequency of each word in their calculations for the final scores. The intuition is that two words cannot be considered relevant to a document if their frequency is very low, regardless of the semantic relationship between them. Two words are considered relevant if at least one of them has a high frequency, and if the semantic relation between them is strong. Based on Newton's law of universal gravitation, the authors propose to compute an attraction force between two words w_i and w_j in a document d as follows:

$$\mathrm{f}(w_i, w_j) = \frac{\mathrm{freq}(w_i) \times \mathrm{freq}(w_j)}{\mathrm{Euclidean}(w_i, w_j)^2} \tag{2.32}$$

In Equation 2.32, $\mathrm{freq}(w)$ is the frequency of the word w in d, and $\mathrm{dist}(w_i, w_j)$ is the Euclidean distance of the word embedding vectors for w_i and w_j. It needs to be noted that Equation 2.32 is used to measure a relation between words. For ranking phrases, we need to use the averaged word embeddings for the words contained in a phrase as parameters for Equation 2.32. Subsequently, Wang et al. (2014) compute the probability of two words co-occurring in a document d using the dice coefficient:

$$\mathrm{Dice}(w_i, w_j) = \frac{2 \times \mathrm{freq}(w_i, w_j)}{\mathrm{freq}(w_i) + \mathrm{freq}(w_j)} \tag{2.33}$$

In Equation 2.33, $\mathrm{freq}(w_i, w_j)$ is the co-occurrence frequency of words w_i and w_j in d. Finally the attraction score is computed as the product of the Dice score (interpreted as phraseness) and the attraction force (interpreted as informativeness), having the following equation:

$$\mathrm{Attr}(w_i, w_j) = \mathrm{Dice}(w_i, w_j) \times \mathrm{f}(w_i, w_j) \tag{2.34}$$

The authors tested their method on the Inspec, DUC2001, and SemEval2010 corpora, having obtained an F1 measure at the top 10 candidates of 0.427, 0.269, and 0.136 respectively.

Previous studies have also noted that graph-based keyphrase extraction methods may fail to deliver keyphrases that are representative of the whole document because top ranked candidates tend to be semantically related. There are approaches that attempt to solve this problem by ranking items with an emphasis on diversity. One such approach is GRASSHOPPER, an algorithm proposed by Zhu et al. (2007) that attempts to combine centrality with diversity, providing top ranked items that differ from each other in order to provide a broad coverage of the whole item set. GRASSHOPPER requires as input a graph W, a probability distribution r encoding a prior ranking, and a weight λ balancing the two, which is equivalent to the damping factor d in the PageRank equation. If no prior ranking is given, the initial probability distribution matrix r assumes that each node has a probability $\frac{1}{|V|}$. The algorithm then extracts the first candidate by performing a random walk similar to that from PageRank, with the transition probability between connected states defined by λ and W, and the probability of randomly jumping (or teleporting) to another state defined by r. Because a stationary probability distribution does not address diversity, only the first candidate is extracted using this approach. This first candidate is then selected using the PageRank metric detailed in Equation 2.19.

After computing the first candidate (i.e., the top ranked node according to PageRank), the corresponding node is then transformed into an absorbing node (i.e., a node g with a transition probability to all other nodes of zero). The problem now becomes computing the expected number of visits to each node before absorption. The intuition is that nodes near the absorbing node g_1 will have fewer visits because the walk will be absorbed shortly after visiting them. Groups of nodes farther from g_1 allow the random walk to linger between them, having more visits. The algorithm then selects the second item g_2, as the node with the largest expected number of visits (i.e., the state with the new highest PageRank score), inhibiting similar items from being selected, and encouraging diversity. The node g_2 is then transformed into an absorbing state, and the procedure is repeated until all items are ranked. The authors suggest further improving the algorithm, using partial absorption by tuning an escape probability, and creating a continuum between PageRank and GRASSHOPPER.

2.2.2 Language Modeling Techniques for Keyphrase Extraction

Statistical language models can also be used as a tool for unsupervised keyphrase extraction (Tomokiyo and Hurst, 2004). According to this methodology, every sequence of words $W = w_1 w_2 w_3 \ldots w_m$ can be assigned a probability based on the number of occurrences of W in a training corpus. If we assume that the probability of any word in the sequence depends only on the n previous words (i.e., a Markov property), then the probability of W can be calculated using n-gram models. In the general case, the probability computation for an n-gram model is computed as follows:

$$\begin{aligned} \mathrm{P}(W) &= \prod_{i=1}^{n} \mathrm{P}(w_i \mid w_1, \ldots, w_{i-1}) \\ &\approx \prod_{i=1}^{m} \mathrm{P}(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) \\ &\approx \prod_{i=1}^{m} \frac{\#(w_{i-(n-1)}, \ldots, w_{i-1}, w_i)}{\#(w_{i-(n-1)}, \ldots, w_{i-1})} \\ &\approx \prod_{i=1}^{m} \frac{\#(w_{i-(n-1)}^{i-1}, w_i)}{\#(w_{i-(n-1)}^{i-1})} \end{aligned} \tag{2.35}$$

In Equation 2.35, #() represents a count function (i.e., the count of occurrences of a given n-gram in a corpus). The simplest case for an n-gram model would be the unigram model corresponding to the following equation:

$$\mathrm{P}(W) \approx \prod_{i=1}^{m} \mathrm{P}(w_i) \tag{2.36}$$

Having access to a background and a foreground corpus, we can create language models for both corpora in order to measure phraseness (i.e., a notion of how much a sequence of words can be considered as phrase) and informativeness (i.e., how well the phrase captures the key ideas in a corpus). We denote the different language models that are involved as: LM_{fg}^{1} for the foreground unigram model, LM_{bg}^{1} for a background unigram model, LM_{fg}^{N} and LM_{bg}^{N} for a foreground and background N-th order language models respectively.

Tomokiyo and Hurst (2004) proposed the use of the pointwise Kullback-Leibler (KL) divergence $\delta_W(\mathrm{p} \parallel \mathrm{q})$ between language models as a way to measure phraseness and informativeness. Let $\mathrm{p}(W)$ and $\mathrm{q}(W)$ be two probability mass functions given by language models. The pointwise KL divergence between them can be defined as follows:

$$\delta_W(\mathrm{p} \parallel \mathrm{q}) = \mathrm{p}(W) \times \log\left(\frac{\mathrm{p}(W)}{\mathrm{q}(W)}\right) \tag{2.37}$$

Phraseness can be quantified as the loss of information by assuming the independence of each word in the unigram model:

$$\delta_W(LM^N_{\text{fg}} \parallel LM^1_{\text{fg}}) \tag{2.38}$$

Informativeness can in turn be defined as how much information we lose by assuming W was extracted from the background model, and it can be computed with either unigram or n-gram models:

$$\delta_W(LM^N_{\text{fg}} \parallel LM^N_{\text{bg}}) \tag{2.39}$$

$$\delta_W(LM^1_{\text{fg}} \parallel LM^1_{\text{bg}}) \tag{2.40}$$

Combining the two ideas results in a mixture of phraseness and informativeness, which can be computed directly as follows:

$$\delta_W(LM^N_{\text{fg}} \parallel LM^1_{\text{bg}}) \tag{2.41}$$

Equation 2.41 gives us a unified score for phraseness and informativeness, but it is also possible to calculate the scores independently, and then combining them by addition or multiplication.

A fundamental task in the context of the proposal from Tomokiyo and Hurst (2004) relates to estimating the parameters of language models from data, with appropriate parameter smoothing. A widely adopted method for parameter smoothing is the Kneser-Ney technique (Kneser and Ney, 1995). In order to understand the Kneser-Ney smoothing technique we first need to present absolute discounting interpolation (ADI). In ADI, a fixed value d is discounted from all counts in order to distribute probability mass to lower order n-gram models. Using a bigram model as example, the absolute discounted probability of a bigram $(w_i w_{i-1})$ can be computed as follows:

$$\text{P}_{\text{AD}}(w_i \mid w_{i-1}) = \frac{\max(\#(w_{i-1}, w_i) - d)}{\#(w_{i-1})} + \lambda(w_{i-1}) \times \text{P}(w_i) \tag{2.42}$$

In Equation 2.42, $\lambda(w_{i-1})$ is an interpolation weight which will be addressed later, and $\text{P}(w_i)$ is the unigram probability, eventually smoothed through a simpler approach such as Laplace Smoothing (i.e., $\text{P}(w_i) = \frac{\#(w_i+1)}{|V|+|dictionary|}$). The Kneser-Ney approach keeps the same interpolation weight, but replaces the estimate for the probabilities of the lower-order unigrams with a continuation probability (i.e., how likely is the word w_i to appear as novel continuation). The continuation probability of a word w can be computed as follows:

$$\mathrm{P}_{\mathrm{Cont}}(w_i) = \frac{|\{w' : 0 < \#(w', w_i)\}|}{|\{(w', w'') : 0 < \#(w', w'')\}|} \tag{2.43}$$

In Equation 2.43, the numerator is the cardinality of the set that contains all bigrams (w', w_i) (i.e., how many words w' appear before w_i), and the denominator is a normalization factor that corresponds to the total number of different word bigrams. The interpolation weight $\lambda(w_{i-1})$ in Equation 2.42 is a normalization constant (i.e., the discounted probability mass), that can be computed as follows:

$$\lambda(w_{i-1}) = \frac{d}{\sum_{w'} \#(w_{i-1}, w')} \times |\{w' : 0 < \#(w_{i-1}, w')\}| \tag{2.44}$$

In equation 2.44, the first term is the normalized discount, and the second term is the number of times we applied the normalized discount (i.e., the number of words w' that can follow w_{i-1}). Substituting in Equation 2.42 the continuation probability and the normalization factor, we can now compute the Kneser-Ney probability of a bigram (w_i, w_{i-1}) as follows:

$$\mathrm{P}_{\mathrm{KN}}(w_i \mid w_{i-1}) = \frac{\max(\#(w_{i-1}, w_i) - d, 0)}{\sum_{w'} \#(w_{i-1}, w')} + \lambda(w_{i-1}) \times \mathrm{P}_{\mathrm{Cont}}(w_i) \tag{2.45}$$

The generalized recursive formula for higher order n-grams, as derived by Chen and Goodmam (1999), is the following:

$$\mathrm{P}_{\mathrm{KN}}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\max(\#(w_{i-n+1}^{i-1}, w_i) - d, 0)}{\sum_{w'} \#(w_{i-n+1}^{i-1}, w')} + \lambda(w_{i-n+1}^{i-1}) \times \mathrm{P}_{\mathrm{KN}}(w_i \mid w_{i-n+2}^{i-1}) \tag{2.46}$$

One important detail that is not obvious in Equation 2.46 is that the first term is only used when $\mathrm{C}(w_{i-n+1}^{i-1}) > 0$. Furthermore, for the lowest order n-gram model, we can use a uniform distribution to prevent divisions by zero. Having $|V|$ as the size of the vocabulary, the 0th-order probability for an n-gram w_i is computed as follows:

$$\mathrm{P}_{\mathrm{KN}}(w_i) = \frac{1}{|V|} \tag{2.47}$$

Chen and Goodmam (1999) proposed a modification to Kneser-Ney smoothing in which the fixed discount d, is replaced with three new parameters, D_1, D_2, and D_{3+}, that are applied to n-grams with counts of one, two, and three or more respectively. The motivation for using different weights is that the ideal average discount for n-grams with higher counts is substantially different than the ideal average discount for n-grams with one or two counts (Chen and Goodmam, 1999). The authors also propose that instead of using Equation 2.46 we use the following equation:

$$\mathrm{P_{KN}}(w_i \mid w_{i-n+1}^{i-1}) = \frac{\#(w_{i-n+1}^{i}) - \mathrm{D}(\#(w_{i-n+1}^{i}))}{\sum_{w_i} \#(w_{i-n+1}^{i})} + \lambda(w_{i-n+1}^{i-1}) \times \mathrm{P_{KN}}(w_i \mid w_{i-n+2}^{i-1}) \tag{2.48}$$

In Equation 2.48, function $\mathrm{D}(x)$ is computed as follows:

$$\mathrm{D}(x) = \begin{cases} 0 & \text{if } x = 0 \\ D_1 & \text{if } x = 1 \\ D_2 & \text{if } x = 2 \\ D_{3+} & \text{if } x \geq 3 \end{cases} \tag{2.49}$$

To make the distribution sum to 1, λ is now computed using the following equation:

$$\lambda(w_{i-n+1}^{i-1}) = \frac{D_1 \mathrm{N}_1(w_{i-n+1}^{i-1}, w') + D_2 \mathrm{N}_2(w_{i-n+1}^{i-1}, w') + D_{3+} \mathrm{N}_{3+}(w_{i-n+1}^{i-1}, w')}{\sum_{w_i} \#(w_{i-n+1}^{i})} \tag{2.50}$$

In Equation 2.50, N_n is the number of unique words w' that follow w_{i-n+1}^{i-1}, with a count of n. It can be formally defined as follows:

$$\mathrm{N}_n(w_{i-n+1}^{i-1}, w') = |\{w' : n = \mathrm{C}(w_{i-1}, w')\}| \tag{2.51}$$

The optimal discount $D_n$values are computed as follows:

$$D_1 = 1 - 2 \times \frac{n_1}{n_1 + 2n_2} \times \frac{n_2}{n_1} \tag{2.52}$$

$$D_2 = 2 - 3 \times \frac{n_1}{n_1 + 2n_2} \times \frac{n_3}{n_2} \tag{2.53}$$

$$D_{3+} = 3 - 4 \times \frac{n_1}{n_1 + 2n_2} \times \frac{n_3}{n_4} \tag{2.54}$$

In the previous equations, n_x is the number of n-grams with a count of x exactly.

Based on the same motivation provided Chen and Goodmam (1999) for extending the Kneser-Ney smoothing method, Shareghi et al. (2016) proposed to further extend the method by accommodating 7 additional discount parameters. As such, $\mathrm{D}(x)$ is now computed as $\mathrm{D}(x)$=D_x, and D_x is computed as follows:

$$D_x = \begin{cases} 0 & \text{if } x = 0 \\ \mathrm{x} - (\mathrm{x}+1) \times \frac{n_{i+1}}{n_i} \times \frac{n_1}{n_1 + 2n_2} & \text{if x} < 10 \\ 10 - 11 \times \frac{n_{11}}{n_{10}} \times \frac{n_1}{n_1+2n_2} & \text{if x} \geq 10 \end{cases} \tag{2.55}$$

The authors show that expanding the number of discount parameters from 3 to 4 substantially reduced the perplexity on all languages for out-of-domain test sets. For a bigram language model, the extra parameter improved the perplexity by 18 points on their English news-test set, and the use of 10 discount parameters improved the perplexity on the same set by 77 points.

2.3 Overview

This chapter started by detailing some fundamental concepts used in NLP, such as POS tagging, which is transversely used in most methods relating to keyphrase extraction. A sample of these methods was presented in 2.2.1, which started by introducing graph centrality measures, and then described several related works based on those measures, such as TextRank (Mihalcea and Tarau, 2004), WordAttractionRank (Wang et al., 2014), PositionRank (Florescu and Caragea, 2017), ExpandRank (Wan and Xiao, 2008), and GRASSHOPPER (Zhu et al., 2007). The performance of these method is analyzed later in this document in Chapter 4, Section 4.3. Section 2.2.2 details alternatives to graph-base centrality measures that make use of statistical models. The latter section also details techniques to estimate parameters of language models from data, using the Kneser-Ney smoothing technique, along with some variations of that method.

3 A Hybrid Method for Keyphrase Extraction

This chapter details a new method for keyphrase extraction combining language modeling techniques, such as the ones proposed by Tomokiyo and Hurst (2004), with a weighted PageRank centrality measure (Page et al., 1999), as a way to estimate the importance of a candidate keyphrases in a given text. This new hybrid method can be divided into three main steps, namely candidate extraction, initial ranking of candidates keyphrases, and candidate re-ranking, these three steps are described in Sections 3.1, 3.2, and 3.3 respectively.

3.1 Candidate Selection

In order to score potential candidates we must first create a graph representation of the text from which we are trying to extract relevant keyphrases. To do so, we must first identify potential candidate phrases contained in the target document. Because keyphrases are typically noun phrases, we can reliably identify candidates using a syntactic filter based on POS tags. Candidates passing this filter will correspond to vertices in the graph representation of the text. Furthermore, in this step we also create the language models, and compute the phraseness and informativeness of each valid candidate. The general procedure for extracting valid candidates is as follows:

1. Tokenize the text into sentences, and further split these sentences sentences by commas, semicolons, parenthesis and quotation marks, thus creating clauses in which we can assume there is a semantic relation between candidates (i.e., multi word candidates will be limited to the words contained within each clause they appear in).

2. Label all words with POS tags using the tagger provided by NLTK python library[1], which currently uses a Perceptron based tagger, and convert all words to lowercase.

3. Generate n-grams within the clauses created in Item 1, having n ranging from 1 to 5.

[1]http://www.nltk.org/

4. Filter the n-grams by removing those that do not match the syntactic pattern `(NN|JJ)*NN`, i.e., valid n-grams must start with a noun (`NN`) or an adjective(`JJ`), and end with a noun.

5. Remove n-grams that pass the POS filter but contain invalid characters such as punctuation marks, mathematical symbols, or other undesired tokens.

It is worth noting that this process for candidate extraction can easily be generalized to other languages for which we have a POS tagger, being that it relies only on POS tagging. We provide an example on how the candidate selection would work on the following text:

Excerpt of a text from SemEval2010

> Efficient discovery of grid services is essential for the success of grid computing. The standardization of grids based on web services has resulted in the need for scalable web service discovery mechanisms to be deployed in grids. Even though UDDI has been the de facto industry standard for web-services discovery, imposed requirements of tight-replication among registries and lack of autonomous control has severely hindered its widespread deployment and usage. With the advent of grid computing the scalability issue of UDDI will become a roadblock that will prevent its deployment in grids. In this paper we present our distributed web-service discovery architecture, called DUDE (Distributed UDDI Deployment Engine). DUDE leverages DHT (Distributed Hash Tables) as a rendezvous mechanism between multiple UDDI registries. DUDE enables consumers to query multiple registries, still at the same time allowing organizations to have autonomous control over their registries. Based on preliminary prototype on PlanetLab, we believe that DUDE architecture can support effective distribution of UDDI registries thereby making UDDI more robust and also addressing its scaling issues. Furthermore, The DUDE architecture for scalable distribution can be applied beyond UDDI to any Grid Service Discovery mechanism.

Following the steps in Items 1 and 2 would produce the following list, where each item is a clause, and POS labels are shown in parenthesis:

- efficient(`JJ`) discovery(`NN`) of(`IN`) grid(`NN`) services(`NNS`) is(`VBZ`) essential(`JJ`) for(`IN`) the(`DT`) success(`NN`) of(`IN`) grid(`JJ`) computing(`NN`)]

- the(`DT`) standardization(`NN`) of(`IN`) grids(`NNS`) based(`VBN`) on(`IN`) web(`NN`) services(`NNS`) has(`VBZ`) resulted(`VBN`) in(`IN`) the(`DT`) need(`NN`) for(`IN`) scalable(`JJ`) web(`JJ`) service(`NN`) discovery(`NN`) mechanisms(`NN`) to(`TO`) be(`VB`) deployed(`VBN`) in(`IN`) grids(`NNS`)

- even(RB) though(IN) uddi(NNP) has(VBZ) been(VBN) the(DT) de(FW) facto(FW) industry(NN) standard(NN) for(IN) web-services(NNS) discovery(NN)
- imposed(JJ) requirements(NNS) of(IN) tight-replication(NN) among(IN) registries(NNS) and(CC) lack(NN) of(IN) autonomous(JJ) control(NN) has(VBZ) severely(RB) hindered(VBN) its(PRP) widespread(JJ) deployment(NN)
- and(CC) usage(NN)
- with(IN) the(DT) advent(NN) of(IN) grid(JJ) computing(VBG) the(DT) scalability(NN) issue(NN) of(IN) uddi(NNP) will(MD) become(VB) a(DT) roadblock(NN) that(WDT) will(MD) prevent(VB) its(PRP) deployment(NN) in(IN) grids(NNS)
- in(IN) this(DT) paper(NN) we(PRP) present(VBD) our(PRP) distributed(JJ) web-service(JJ) discovery(NN) architecture(NN)
- called(VBN) dude(NNP)
- distributed(NNP) uddi(NNP) deployment(NNP) engine(NNP)
- dude(NNP) leverages(VBZ) dht(NNP)
- distributed(NNP) hash(NNP) tables(NNP)
- as(IN) a(DT) rendezvous(JJ) mechanism(NN) between(IN) multiple(JJ) uddi(NNP) registries(NNS)
- dude(NNP) enables(VBZ) consumers(NNS) to(TO) query(VB) multiple(JJ) registries(NNS)
- still(RB) at(IN) the(DT) same(JJ) time(NN) allowing(VBG) organizations(NNS) to(TO) have(VB) autonomous(JJ) control(NN) over(IN) their(PRP) registries
- based(VBN) on(IN) preliminary(JJ) prototype(NN) on(IN) planetlab(NNP)
- we(PRP) believe(VBP) that(IN) dude(NNP) architecture(NN) can(MD) support(VB) effective(JJ) distribution(NN) of(IN) uddi(NNP) registries(NNS) thereby(RB) making(VBG) uddi(NNP) more(JJR) robust(JJ) and(CC) also(RB) addressing(VBG) its(PRP) scaling(NN) issues(NNS)
- furthermore(RB)
- the(DT) dude(NNP) architecture(NN) for(IN) scalable(JJ) distribution(NN) can(MD) be(VB) applied(VBN) beyond(IN) uddi(NNP) to(TO) any(DT) grid(NNP) service(NNP) discovery(NNP) mechanism(NN)

Finally, Items 3 to 5 would generate the following 143 candidates: *web service,lack*, *grid computing*, *paper*, *efficient discovery*, *consumers to query multiple registries*, *issues*, *web*, *mechanism between multiple uddi*, *deployment and usage*, *distributed*, *need for scalable web service*, *web service discovery mechanisms*, *uddi registries*, *autonomous control*, *rendezvous mechanism*, *distributed uddi deployment engine*, *widespread deployment and usage*, *efficient discovery of grid services*, *grid*, *deployment*, *grids based on web*, *standardization*, *success*, *advent*, *grid services*, *roadblock*, *architecture*, *scalability issue of uddi*, *registries*, *same time allowing organizations*, *grids*, *grid service discovery mechanism*, *grid service discovery*, *web-services*, *effective distribution of uddi*, *distributed hash tables*, *deployment engine*, *effective distribution of uddi registries*, *consumers*, *service discovery mechanism*, *preliminary prototype on planetlab*, *distributed uddi deployment*, *web services*, *time allowing organizations*, *issue*, *discovery mechanism*, *grid computing the scalability issue*, *industry standard*, *scaling issues*, *hash*, *tables*, *scalable web service discovery*, *mechanisms*, *scalability*, *standard*, *industry standard for web-services discovery*, *rendezvous mechanism between multiple uddi*, *scalable web service discovery mechanisms*, *dude architecture*, *preliminary prototype*, *dude architecture for scalable distribution*, *registries and lack*, *distributed uddi*, *industry*, *tight-replication among registries and lack*, *distribution of uddi*, *widespread deployment*, *service discovery mechanisms*, *distributed web-service discovery*, *discovery architecture*, *imposed requirements of tight-replication*, *scalability issue*, *requirements of tight-replication among registries*, *tight-replication among registries*, *grids based on web services*, *dude enables consumers*, *service*, *standardization of grids*, *distributed web-service discovery architecture*, *lack of autonomous control*, *dude*, *uddi deployment*, *dude leverages dht*, *architecture can support effective distribution*, *prototype*, *organizations to have autonomous control*, *multiple uddi registries*, *engine*, *web service discovery*, *usage*, *uddi*, *architecture for scalable distribution*, *uddi deployment engine*, *multiple registries*, *uddi registries thereby making uddi*, *discovery of grid services*, *organizations*, *mechanism between multiple uddi registries*, *requirements of tight-replication*, *web-service discovery*, *efficient discovery of grid*, *distribution*, *uddi to any grid service*, *control*, *computing*, *hash tables*, *discovery*, *tight-replication*, *uddi to any grid*, *need*, *service discovery*, *scalable web service*, *same time*, *requirements*, *distributed hash*, *standard for web-services*, *discovery of grid*, *discovery mechanisms*, *mechanism*, *dht*, *multiple uddi*, *deployment in grids*, *web-services discovery*, *planetlab*, *prototype on planetlab*, *success of grid computing*, *uddi will become a roadblock*, *services*, *industry standard for web-services*, *scaling*, *imposed requirements*, *grid computing the scalability*, *web-service discovery architecture*, *grid service*, *distribution of uddi registries*, *registries thereby making uddi*, *time*, *standard for web-services discovery*, *scalable distribution*, *effective distribution*, *issue of uddi.*

3.2 Initial Ranking

After extracting valid candidates, we perform an initial ranking of all candidates by computing the TF-IDF score of all the valid candidates (i.e., n-grams that pass the filters described in section 3.1, Items 4 and 5.

Following the initial ranking of the candidates through their TF-IDF score, we compile a background corpus (i.e., a compilation of all the texts in the corpus) and a foreground corpus (i.e., the target text), and create Knesser-Ney smoothed unigram and 5-gram language models from both the background and the foreground corpus. These language models can, in turn, be created using the KenLM Language Modelling Toolkit[2] (Heafield et al., 2013). Although we will use these language models to compute a prior probability for each candidate keyphrase, an alternative application would be to replace the initial ranking using the TF-IDF scores, with a combination of the phraseness and informativeness scores of each candidate.

Having performed these steps we now have a list of n-grams which we considered to be valid candidates, initially scored according to their TF-IDF, and that will now be re-rank as described in Section 3.3.

3.3 Re-Ranking

Following the initial ranking of candidates, we then create a graph representation of the text. To do so, we add all valid candidates with a TF-IDF score above a certain threshold as vertices to a graph. This threshold has been set to the top 900 candidates in order to keep the graph size manageable, because, as we can see from the example provided in the end of Section 3.1, even a small excerpt of text generates a large number of candidate keyphrases. The threshold is a parameter that can be adjusted if, as an example, the corpus from which we are extracting keyphrases is composed of smaller texts, for which, this threshold would allow most candidates to be included in the graph. Each vertex p is then given a prior weight computed as the weighted sum of the phraseness and informativeness score of each candidate, using Equations 2.38 and 2.39 respectively. The probabilities for each candidates, needed to compute the phraseness and informativeness of each candidate, are given by the language models described previously in Section 2.2.2. Assuming that wp is a parameter that defines how much weight we want to give to both the phraseness and the informativeness value of each candidate in the computation of

[2]https://kheafield.com/code/kenlm/

their prior, then the prior weight Prior(p) of a candidate p is computed as follows:

$$\text{Prior}(p) = wp \times \text{Phraseness}(p) + (1 - wp) \times \text{Informativeness}(p) \tag{3.1}$$

In Equation 3.1, the wp parameter allows us to control what we value most in the prior weight of each candidate, we can consider only the phraseness of each candidate by setting $wp = 1$, we can consider only the informativeness of a candidate by setting $wp = 0$, or any combination of both. Additionally, two other parameters are used to modify the prior probabilities of the candidates. With these parameters we can take into account both the position of the candidate keyphrase in the text from which we are extracting keyphrases, and the number of words in the candidate. Assuming both of these parameter are true, then we compute the modified prior of each candidate as follows:

$$\text{MPrior}(p) = \text{Prior}(p) \times \left(\frac{1}{20 + line}\right) \times \left(\frac{1}{e^{|(2-|p|)|}}\right) \tag{3.2}$$

In Equation 3.2, $line$ is the position of the first sentence in which the candidate p appears in the text, and $|p|$ is the number of words in p. Thus, the second term takes into account the position of the candidate. Although this term reduces the Prior of all the candidates, those that appear sooner in a text will have their Prior reduced less than candidates that appear towards the end of the text. The intuition is that candidates that appear towards the beginning of the text tend to be more relevant, a good example would be candidates that appear in the title since these are very likely to be keyphrases. Furthermore, this reduction in prior is attenuated so that candidates that appear in the beginning of the text and in close proximity, will not have a large difference in the reduction of their Prior, and candidates towards the end of the text will have a negligible difference between their attenuation. The third term takes into account account the length of the candidate, favoring candidates of length 2, since human annotators more often pick candidates of length 2 as keyphrases (Rousseau and Vazirgiannis, 2015), followed by candidates of length 1 and 3. Prior(p) is computed using Equation 3.1. An alternative way of computing the prior of a candidate was incorporated into the method, such that, the prior can also be computed using the PositionRank approach described in Section 2.2.1, with a slight modification because the method proposed in this project extracts multi-word candidates. This modification is simply to consider the position of any n-gram as being the position of the first word it contains. Due to it lowering the performance of the new method, this alternative was disregarded.

After computing the prior weights of every candidate, weighted edges are added between candidates if they co-exist within a sentence. The weight of these edges attempts to capture semantic similarities between candidates. The goal is that candidates would, in a PageRank centrality measure, share a higher amount of their weight with other candidates whose words represent similar concepts. As such, their computation is based on the following factors: the distance between vector representations of two candidates, the number of times these candidates co-occur in a sentence, and the distance between these candidates in each co-occurrence. The vector representation of each candidate is the average of the pre-trained GloVe (Pennington et al., 2014) word embedding[3] of each word in the candidate, weighted by the document frequency of that word. Assuming that $\mathbf{e_i}$ represents the pre-trained word embedding of a word w_i in a candidate $p = w_1 w_2 ... w_n$, then the vector representation $\mathbf{V}_p$ of a candidate p is computed as follows:

$$\mathbf{V}_p = \frac{\sum_{i=1}^{n} \text{idfs}(w_i) \times \mathbf{e}_i}{\sum_{i=1}^{n} \text{idfs}(w_i)} \tag{3.3}$$

In Equation 3.3, $\text{idfs}(w_i)$ is a value proportional to the document frequency of the word w_i and the number of documents in the corpus that contains the document from which we are trying to extract keyphrases. Let $|Corpus|$ be the number of documents in the corpus, and let $\text{dfs}(w_i)$ be the number of documents in which w_i occurs, then $\text{idfs}(w_i)$ is computed as follows:

$$\text{idfs}(w_i) = 50 + \log \left(\frac{|Corpus|}{\text{dfs}(w_i)} \right) \tag{3.4}$$

We then create a matrix $\mathbf{M}$ containing all the vectors $\mathbf{V_p}$ of all the valid candidates in the corpus. This matrix allows for the creation of a feature vector $\mathbf{F_p}$ for each candidate, which is computed by extracting the 35 most significant dimensions through a Singular Value Decomposition (SVD) on $\mathbf{M}$. We perform SVD because, as it was shown by Levy et al. (2015), it is the best performing method for at similarity tasks. A measure of similarity $\text{Sim}(p_i, p_j, S)$ can now be computed between two candidates p_i and p_j, within a given sentence S. This measure will depend on the feature vectors of the candidates, and on distance between them in the sentence S. Moreover, the computation of the similarity will also depend on whether one candidate is nested within the other. The way in which this similarity is computed means that vertices in the graph will give a higher share of their weight to candidates which have a higher semantic similarity to them. Let $\text{Contains}(p_i, p_j)$ be a function that is true if p_i is contained in p_j, and let $\text{Distance}(p_i, p_j, S)$ be the number of words between the candidates p_i and p_j within the sentence S. Then, the similarity measure is computed as follows:

[3]https://nlp.stanford.edu/projects/glove/

$$\mathrm{Sim}(p_i, p_j, S) = \begin{cases} \mathrm{MPrior}(p_i) \times \mathrm{MPrior}(p_j) & \text{if } \mathrm{Contains}(p_i, p_j) \\ \frac{\mathrm{MPrior}(p_i) \times \mathrm{MPrior}(p_j)}{e^{(\mathrm{Euclidean}(\mathbf{F}_{p_i}, \mathbf{F}_{p_j}))}} \times \frac{1}{\log(2+\mathrm{Distance}(p_i, p_j, S))} & \text{otherwise} \end{cases} \tag{3.5}$$

Having defined a formula for the similarity between pairs of candidates we are now able to compute a weight for all the edges in the graph. Let $S = S_1, S_2...S_n$ be the set of sentences in which the candidates p_i and p_j co-occur. Then, the weight of the directed edge from p_i to p_j is computed as follows:

$$\mathrm{Weight}(p_i, p_j) = \sum_{z=1}^{n} \mathrm{Sim}(p_i, p_j, S_z)^{\beta} \tag{3.6}$$

In Equation 3.6, β is a parameter whose value is 1 if $|p_i| \leq |p_j|$ and 2 otherwise. Finally, we compute the score S(p) of each candidate p_i. using a weighted PageRank approach in which the importance of a candidate in a graph depends on a previous weight assigned to it, on the number of directed edges pointing to it, and on the weights of those edges. This method follows a teleporting random walk model, in which a random walker will walk from a node p_i to any node connected to it with a probability proportional to a parameter d, and will teleport to a random node in the graph with a probability proportional to $1 - d$. The score of a candidate represents the amount of time a walker following this approach would spend on each node and is formally computed as follows:

$$\mathrm{S}(p_i) = (1-d) \times \frac{\mathrm{MPrior}(p_i)}{\sum_{p_j} \mathrm{MPrior}(p_j)} + d \times \sum_{p_j \in \mathrm{Links}(p_i)} \frac{\mathrm{S}(p_j) \times \mathrm{Weight}(p_i, p_j)}{\sum\limits_{p_k \in \mathrm{Links}(p_j)} \mathrm{Weight}(p_j, p_k)} \tag{3.7}$$

In Equation 3.7, the parameter d is set to 0.4, and the functions MPrior(p) and Weight(p_i, p_j), are computed thorough Equations 3.2 and 3.6, respectively.

As an additional step, we select the top n candidates with the highest score, and remove candidates contained within others in the top n candidates list, i.e., if a candidate p_i contains p_j, then we remove p_j from the top n candidates list and add the candidate with the top $n + 1$ score to the list. Furthermore, if the removal of a candidate allows for the inclusion in the top n list of another candidate that contains it, then the former is also removed, thus favoring longer candidates. These top n candidate are the keyphrases returned by the hybrid method. Alternatively, an implementation of the GRASSHOPPER algorithm, described in

Section 2.2.1, was included into the method. As such, after ranking all candidates according to Equation 3.7, the top ranked candidate would be removed from the graph, transformed into an absorbing state, and new scores would be computed for all remaining nodes. The process is repeated until 30 candidates are selected, being those that had the highest score at each iteration. These candidates would then go trough the additional post-processed step that removes nested candidates. Since the inclusion of this additional re-ranking of candidates lowered the performance of the new method, this step was disregarded.

The final method has several parameters that can be adjusted to improve its performance such as the the maximum length of the n-grams extracted as candidates from the text. Additionally, we can take into account both the position where the candidates first appear in the text, as well as its length. The TF-IDF threshold can be changed. We can vary the damping parameter, and weight of the phraseness attribute wp of each candidate. To find the combination of parameters that yield the best results, we varied the TF-IDF threshold from 600 to 2000 in increments of 5, and for each increment of the threshold we varied the damping parameter d from 20 to 70 in unitary increments. Finally, for each combination of the threshold and damping parameter, we tested both with and without taking into account the position of the candidate keyphrase. The best combination of these parameters produced the results presented in the following chapter.

3.4 Summary

This chapter details the architecture of the of the proposed keyphrase extraction method. It starts by describing how candidates are extracted form a text in Section 3.1, along with an example to illustrate that process. Section 3.2 shows how the candidates are initially ranked using TF-IDF, in a way that is useful for the subsequent re-ranking stage of the method. It also provides an alternative ranking method that could be used in substitution of the TF-IDF heuristic. Section 3.3 detail the process of computing prior probabilities for each candidate. This Section also describes how to create the graph in order to compute a centrality measure for each candidate; and how the initial ranking process can be helpful to reduce the number of candidates added to the graph, and to remove undesired candidates. Moreover, it shows how the edges are weighed according to the semantic similarity between the candidates, which is done by computing the euclidean distance between the feature vectors of each candidate. These vectors are based on pre-trained word embeddings that provide the semantic representation of each candidate, as well as other features of the candidates such as, their document frequency, the position of that candidate in the text, and it's length. A rationale for the inclusion of these features was also provided.

4 Experimental Evaluation

This chapter presents the experimental evaluation of the proposed keyphrase extraction method, which involved tests with three different corpora commonly used to evaluate keyphrase extraction methods. Section 4.1 presents the general evaluation methodology, together with the description of the evaluation metrics used to measure the performance of the method. Next, Section 4.2 provides a description of the corpora used in the evaluation, along with any pre-processing that had to be done. Section 4.3 shows the results of the evaluation, how they compare with other related works in keyphrase extraction, and what conclusions can be drawn from these results. Finally, Section 4.4 presents a case study in which a spatial autocorrelation statistic, the Moran's I, is incorporated into the keyphrase extraction method, as well as an analysis of the results of this incorporation.

4.1 Methodology and Evaluation Metrics

The typical accuracy assessment for a keyphrase extraction method involves extracting the candidate keyphrases from the target documents, stemming the resulting candidates, and comparing them with the stemmed versions the gold standard keyphrases for that document. The results of the comparison between candidates and gold standard keyphrases can be summarized by several statistics, such as precision, recall, and F1-score.

In classification experiments, precision is a measure of the class agreement between the labels in the data and the labels given by the classifier. It is computed as the ratio between the number of examples correctly classified as true positives TP (i.e., the returned keyphrases that are relevant), and the total number examples labeled by the system as positive (i.e., all keyphrases that are returned).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

In Equation 4.1, FP is the number of examples incorrectly classified as positive (i.e., false positives). Recall measures the effectiveness of a classifier in identifying positive labels. It can be calculated as the number of true positives TP divided by the number of positive examples in the data (i.e., all relevant keyphrases).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

In Equation 4.2, FN is the number of examples incorrectly classified as negative (i.e., false negatives). The F1-score is a particularization of the F-Measure, it is a combination of precision and recall and is computed as follows:

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.3}$$

The metrics described in this section are the ones that will be used to evaluate the performance of the new method.

4.2 Corpora Description

The three corpora used in the evaluation phase were obtained online[1]. A summary of the number of documents and keyphrases in each corpus is present in Table 4.1. The following paragraphs contain a description of each corpus, and all pre-processing steps that were applied to them.

The Inspec dataset is a corpus created by Hulth (2003). It contains 2000 abstracts in English, with their corresponding titles and keyphrases from the Inspec database. Each abstract contains two sets of keyphrases assigned by a professional indexer. The first set is called a controlled set, having keyphrases limited to expressions that appear in the Inspect thesaurus. The second set is called an uncontrolled set and contains keywords that can be any suitable term. In the context of this paper, only the uncontrolled set of keyphrases was considered, and only the test subset containing 500 abstracts was used. Pre-processing of this corpus consisted of removing both tab and return carriage symbols from the beginning of each line.

The SemEval-2010 dataset (Kim et al., 2010) is a corpus containing 244 conference and workshop papers from the ACM Digital Library, partitioned into trial (i.e., a subset of the training data), training and test subsets. The input papers ranged from 6 to 8 pages long and were selected from multiple research areas to ensure a variety of different topics. Keyphrases are divided into sets, one containing author-assigned keyphrases, one containing reader-assigned

[1]https://github.com/snkim/AutomaticKeyphraseExtraction

Corpus	Number of Documents	Number of Keyphrases	Keyphrases per Document
SemEval-2010	100	1534	15
Inspec	500	4913	10
DUC2001	308	2488	8

Table 4.1: Number of documents, keyphrases by corpus, and average number of keyphrases per document.

keyphrases, and one combining the two previous sets. Our method was tested on the test subset containing 100 documents, and no pre-processing of the corpus was done. Results were compared against the combined keyphrase set.

The DUC2001 dataset (Over and Yen, 2001) is a corpus containing 308 news articles with 2488 manually annotated keyphrases, having at most 10 keyphrases per article. Pre-processing of the corpus consisted on the removal of the XML tags, as well as the creation of a new key file for each document containing its gold standard keyphrases, this way adapting and normalizing the original data format.

4.3 Results and Analysis

Performance was measured using the metrics described in Section 4.1, and compared against several related works on keyphrase extraction, such as the TextRank (Mihalcea and Tarau, 2004) and SGRank (Danesh et al., 2015) algorithms. The implementations for these algorithms which were used for comparison is available on the textacy[2] Python library. Performance was measured on the top 5, 10, and 15 highest ranked candidates by each algorithm. It is worth noting that the original implementation of TextRank does not return a fixed number of top candidates, but instead returns the n highest ranking candidates, where n equals one third of the number of total candidates. As such, the version we use for comparison does not follow the exact implementation of the TextRank algorithm given in Section 2.2.1. Furthermore, two baseline implementations of our method were considered for comparison. In the first baseline, candidates are added to the graph as described in Section 3.1, but no previous weights are assigned to the vertices, and no weight is assigned to the edges. The Baseline method is basically an implementation of TextRank with $n$grams as vertices and without collapsing adjacent candidates in post-processing, having the score of each candidate being computed using Equation 3.7, with w_{ij} and Prior(p) set to 1, and having d set to 0.85, (i.e., a PageRank centrality measure with unweighted edges and no

[2]https://github.com/chartbeat-labs/textacy

		DUC2001			Inspec			SemEval		
		5	10	15	5	10	15	5	10	15
TF-IDF	P	0.148	0.139	0.131	0.181	0.165	0.153	0.296	0.260	0.235
	R	0.100	0.137	0.167	0.112	0.149	0.179	0.100	0.126	0.146
	F	0.119	0.138	0.147	0.138	0.156	0.165	0.149	0.170	0.180
Baseline	P	0.111	0.105	0.100	0.112	0.114	0.112	0.134	0.132	0.128
	R	0.072	0.103	0.128	0.071	0.108	0.140	0.047	0.068	0.087
	F	0.087	0.104	0.112	0.087	0.111	0.125	0.070	0.090	0.104
Informativeness	P	0.122	0.109	0.100	0.146	0.147	0.147	0.240	0.226	0.212
	R	0.081	0.106	0.124	0.091	0.137	0.181	0.080	0.112	0.137
	F	0.097	0.107	0.111	0.112	0.141	0.162	0.120	0.150	0.166
Complete	P	0.192	0.166	0.152	**0.314**	**0.291**	**0.273**	**0.426**	**0.372**	**0.334**
	R	0.126	0.154	0.181	**0.191**	**0.253**	**0.304**	**0.142**	**0.179**	**0.205**
	F	0.152	0.160	0.165	**0.238**	**0.271**	**0.288**	**0.213**	**0.241**	**0.254**
TextRank	P	**0.290**	**0.269**	**0.255**	0.295	0.279	0.267	0.062	0.061	0.060
	R	**0.181**	**0.238**	**0.284**	0.182	0.247	0.297	0.020	0.030	0.040
	F	**0.233**	**0.253**	**0.268**	0.225	0.262	0.281	0.030	0.040	0.048
SGRank	P	0.212	0.189	0.174	0.307	0.283	0.264	0.310	0.268	0.238
	R	0.137	0.176	0.209	0.190	0.246	0.289	0.105	0.129	0.146
	F	0.166	0.183	0.190	0.235	0.263	0.276	0.157	0.174	0.181

Table 4.2: Performance of different methods at the 5, 10, and 15 top ranked candidates.

prior weights).

Table 4.2 shows the performance of our method on the different datasets. The Baseline and Complete lines correspond to our baseline and to the complete algorithms, respectively. In the second baseline, the Informativeness method, the score of a candidate is given by its informativeness score, setting the weight wp in Equation 3.1 to 0. Because the implementations by the textacy library are not able to reproduce the results presented by the authors of the different methods in their original papers, we included in Table 4.3 the original results presented by the authors on the corpora described in this section. Some data in Table 4.3 is annotated, this means that the results at that particular cell were not presented by the authors, but were instead extracted from the papers presented by: 2) Danesh et al. (2015), 3) Wang et al. (2014), 4) Hasan and Ng (2014). Results annotated with 1) mean that the method did not return the number of keyphrases on that column, but instead returned the 30% highest ranked candidates. Furthermore, Table 4.3 contains the performance of other relevant methods for keyphrase extraction, whose implementations were not reproduced during the realization of this project.

The results obtained from the testing show that the new method proposed in this paper is effective at extracting keyphrases, being comparable to other state-of-the art methods. Furthermore, we can make several noteworthy observations from Table 4.2. First, that the Baseline method outperforms TextRank in the SemEval dataset. This indicates that a previous selection of multi-word candidates as vertices for the graph can be more advantageous than selecting just individual words as potential candidate keyphrases. However, this appears to be corpus

	DUC2001			Inspec			SemEval		
	5	10	15	5	10	15	5	10	15
TF-IDF (Kim et al., 2010)	–	0.263$^{(3)}$	0.270$^{(4)}$	–	–	0.363$^{(4)}$	0.112	0.144	0.151
HUMB (Lopez and Romary, 2010)	–	–	–	–	–	–	0.198	0.259	0.275
SGRank (Danesh et al., 2015)	–	–	–	**0.296**	0.339	**0.336**	**0.260**	**0.272**	**0.281**
TopicRank (Bougouin et al., 2013)	–	–	–	–	0.279	–	–	0.121	–
TextRank (Mihalcea and Tarau, 2004)	–	0.102$^{(1)}$	–	0.235$^{(2)}$	0.306$^{(2)}$/0.362$^{(1)}$	0.279$^{(2)}$	0.012$^{(2)}$	0.024$^{(2)}$	0.034$^{(2)}$
WordAttractionRank (Wang et al., 2014)	–	0.269	**0.277**	–	**0.427**	–	–	–	0.136
ExpandRank (Wan and Xiao, 2008)	–	**0.317**	–	–	0.398$^{(3)}$	–	–	–	0.035$^{(3)}$
SingleRank (Wan and Xiao, 2008)	–	0.272	–	–	0.398$^{(3)}$	–	–	–	0.035$^{(3)}$
Complete	0.152	0.160	0.165	0.238	0.271	0.288	0.213	0.241	0.254

Table 4.3: F1 metric reported in the literature for different methods at 5, 10, and 15 top ranked candidates.

dependent, seeing that the usage of single word candidates by TextRank in the ranking phase, outperforms the Baseline method in the Inspec and DUC2001 corpora. Second, that the selection of multi-word candidates, and the additional step described in the end of Section 3.3 that favours longer candidates, improves the TF-IDF baseline presented by Kim et al. (2010). Moreover, the Informativeness method only takes into account the informativeness of a candidate, because, when increasing the phraseness contribution in both the Informativeness and Complete methods (i.e., increasing wf in Equation 3.1), their performance lowered. This suggests that the notion of phraseness (i.e., how much can a sequence of words be considered a phrase), is well captured by the usage of a syntactic filter. Finally, that the inclusion of weighted edges and a prior probability for every candidate, increased the performance of the Baseline method in all corpora, (i.e., the Complete method outperforms the Baseline method in all corpora).

From the results presented in the literature, shown in Table 4.3, we can see that the method proposed in this paper is not enough to surpass several state-of-the art methods in multiple corpora. Analyzing the results pertaining to the SemEval2010 corpus, if we consider the HUMB method, which won the SemEval-2010 Task 5, the Complete method only surpasses its F1 metric at the top 5 keyphrases. It performs worse at the top 10 and 15 keyphrases because HUMB exceeds its recall at both of those marks, with HUMB scoring a recall of 0.218 at 10, and 0.278 at 15. Nonetheless the Complete method outperforms HUMB at 10 and 15 in the precision metric, with HUMB scoring a precision of 0.320 at 10, and 0.272 at 15. One reason may be that due to the usage of a syntactic filter and TF-IDF threshold, the Complete method excludes some keyphrase that HUMB is able to identify. The same comparison if more difficult to be done with SGRank, since Danesh et al. (2015) do not report precision and recall at the same number of keyphrases. Additionally, since testing could not replicate the original results of SGRank, there could be some pre-processing of the corpus that would bring the results their method obtained in the testing phase of this project, to the level the authors obtained in their original paper. If that is the case, then that pre-processing could also help improve the Complete method.

When comparing the results of the Complete method in the Inspec dataset, one can see that it trails behind the highest ranking method, the WordAttractionRank. Looking at the implementation described by Wang et al. (2014), we can identify some of the same heuristics used in the Complete method, such as using word embeddings as a basis for weighing the edges of a graph representation of the text, and a syntactic filter based on POS labels. A significant difference in WordAttractionRank is the length of the candidates that are chosen, being that only unigrams are selected as vertices for the graph, and the generation of multi-word candidates is done by collapsing adjacent candidates whose score is above a certain threshold (i.e., the same process Mihalcea and Tarau (2004) use in their approach). This difference indicates that it is better, for corpus containing smaller texts, to follow their approach for generating candidates. Another observation that supports the previous assertion is that the performance of WordAttractionRank drops significantly in the SemEval2010 corpus which consists of larger scientific papers while the Inspec corpus is a collection of abstracts. However, SGRank has a higher performance in the Inspec corpus than the SemEval2010, and because SGRank extracts multiple word candidates as candidate keyphrases, this may contradict the assertion that selecting unigrams as candidates is a better approach for small corpus. One factor that could explain this is the co-occurrence window used in SGRank, the window is unusually high, which may have a more significant impact than the way candidates are generated.

The major difference between the two highest scoring approaches in the DUC2001, the WordAttractionRank and the ExpandRank, is the way in which the edges are weighted in graph. While ExpandRank uses co-occurrence frequencies calculated from both the document itself and the neighbour documents (i.e., documents which their algorithms classifies as similar), WordAttractionRank computes similarity based on statistics about the candidates and the distance between their word embeddings. The Complete method uses both of those approaches, as such, more testing would need to be done to see if excluding one of those approaches would improve the results of the Complete method.

It is worth noting that the parameters of the Complete method were tuned in the SemEval2010 dataset, this means that the results presented in Table 4.3, may not be the highest scores the Complete method could attain in both the Inspec and DUC2001 corpora.

4.4 Case Study in Geo-Temporal Characterization of Candidate Keyphrases

One of the aims of this project was to assert the usefulness of incorporating spatial autocorrelation metrics into the keyphrase extraction algorithm. The usage of a spatial autocorrelation metrics may allow us to identify keyphrases which are important to a given geographic region by taking into account their geographic uniqueness. As an example, if we were studying political transcripts from a given municipality, it would be interesting to identify which phrases are geographically correlated to that municipality, since those phrases may be topics of interest for that particular region. Furthermore, if a phrase that is typically not found in a particular region, arises in a text from that region, then its usage may indicate that it is an important phrase for that text since its is not usually used in the literature from that region. We chose to incorporate a commonly used metric for spatial autocorrelation: the Moran's I statistic (Moran, 1950). This statistic measures whether the frequency of an observation, be it words, phrases, temperature, population density, or any other observable data, is more probable in proximity to other similar observations (i.e., does the number of observation on a given location correlate to the number of observations in neighbouring locations). The Moran's I statistic has a value between -1 and 1, being that an observation with a Moran's I value of 1 is considered to have a high positive spatial autocorrelation (i.e., the number of observations in a given region tends to be similar to the number of observations in neighbouring regions), a value of -1 indicates a negative spatial autocorrelation (i.e., the number of observations in a given region tends to differ from the number of observations in the neighbouring regions), and a Moran's I of 0 indicates that the observations have no spatial autocorrelation. To define the statistic, let $\mathbf{W} = \{w\}_{i,j \in \{1 \ldots n\}}$ represent a spatial weighting matrix, where larger values of w_{ij} indicate greater proximity between the spatial locations i and j, and $w_{ii} = 0$. For a critical threshold τ, Grieve et al. (2011) define $\mathbf{W}$ as follows:

$$w_{ij} = \begin{cases} 1, & d_{ij} < \tau,\ i \neq j \\ 0, & d_{ij} \geq \tau,\ \text{or } i = j \end{cases} \tag{4.4}$$

For this paper, instead of a threshold τ, we chose to compute $\mathbf{W}$ using edge adjacency, having the entry $w_{ij} = 1$ if the municipalities i and j share a border, and 0 otherwise. This matrix was computed using data from a shapefile containing the border coordinates for all 308 municipalities of Portugal as of 2011. If we have a vector $\mathbf{OP} = \{o_{p_1}, \ldots, o_{p_i}\}$, where o_{p_i} represents the number of observations of the keyphrase p in the spatial location i, then the Moran's I statistic of a candidate p is computed as follows:

$$\mathrm{I}(p) = \frac{n}{\sum_i^n (o_{p_i} - \bar{o}_p)} \times \frac{\sum_i^n \sum_j^n w_{ij}(o_{p_i} - \bar{o}_p)(o_{p_j} - \bar{o}_p)}{\sum_i^n \sum_j^n w_{ij}} \tag{4.5}$$

In order to computed the Moran's I statistic of any candidate keyphrase we needed to create a corpus which contains spatial data for each text it contained. This corpus was created by collecting folk tales and urban legends from a web-repository [3] that indicates in which Portuguese municipality the texts were collected. The initial corpus consisted of 3705 HTML files, downloaded automatically from the repository, each one containing a text and the location of its origin. The files were processed to remove all HTML tags, and the text was striped of tabulation marks and return carriages. The final corpus consists of 3705 text files that can be used as input for the new keyphrase extraction algorithm, as well the location in which the texts were collected.

To compute the Moran's I statistic as shown in Equation 4.5, we need the vector of linguistic observations. This was constructed by extracting all viable candidates using the method described in Section 3.1 for each text in the corpus. Candidates were lemmatized using LemPORT (Rodrigues et al., 2014), and the POS-tagging was done using a pre-trained Portuguese model[4] from Google's SyntaxNet (Andor et al., 2016). Since each text is annotated with the municipality it was collected from, the process yielded a list of valid candidates, and the locations and frequencies in which they occur. With this data we were able to compute the Moran's I statistic of every candidate p using Equation 4.5.

Figure 4.1 shows the distributions of occurrences for two candidate phrases. The map on the left shows the occurrences of the phrase *santa maria* with a Moran's I of 0.0052, and the map on the right shows the occurrences of the phrase *amendoeira em flor* with a Moran's I statistic of 0.2920. We can deduce from the Moran's I of each candidate, that the phrase *amendoeira em flor* should have a higher geographic autocorrelation than the phrase *santa maria.* Their distribution on a map reflects this assertion, as seen in Figure 4.1: the occurrences of *amendoeira em flor* are mostly concentrated in the southern municipalities, and the occurrences of the phrase *santa maria* are more evenly dispersed throughout the map.

For each text in the corpus, we extract the most relevant keyphrases using the hybrid method

[3]http://www.lendarium.org/

[4]https://github.com/tensorflow/models/blob/master/syntaxnet/g3doc/universal.md

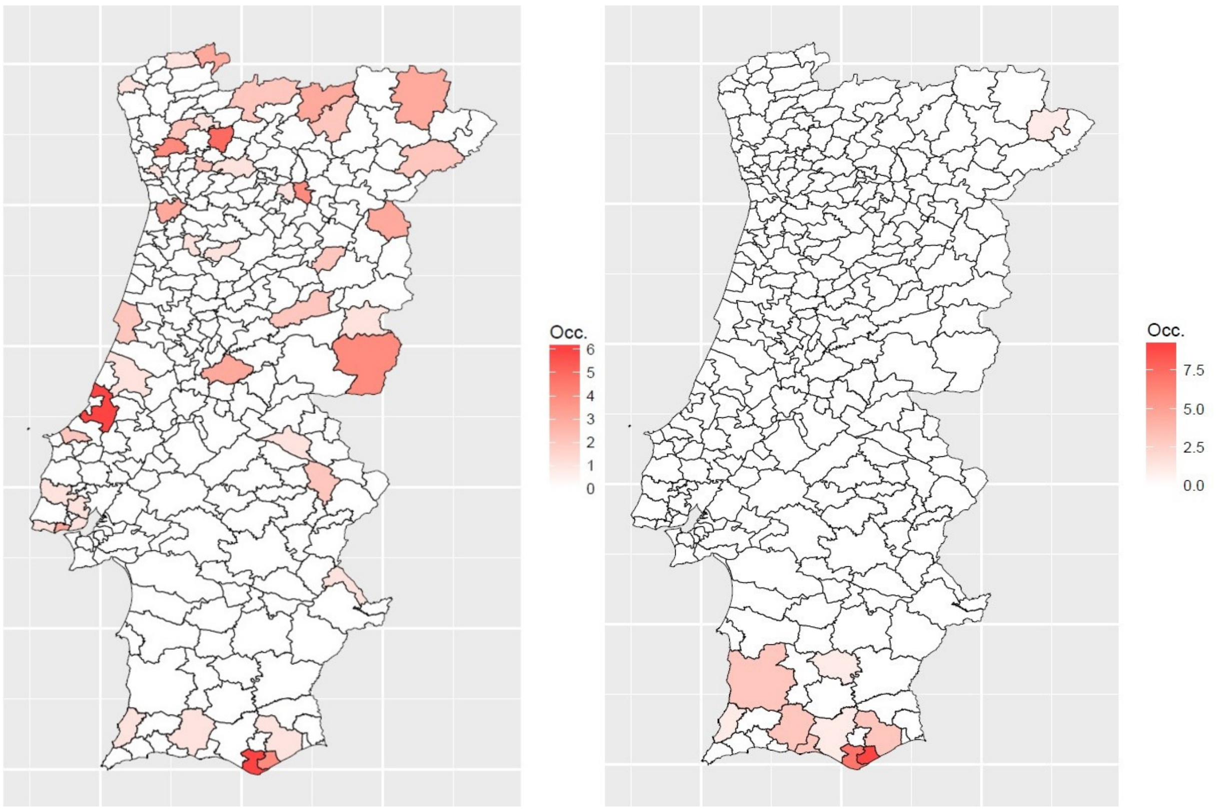

Figure 4.1: Map on the left shows the locations in which the phrase *santa maria* occurs. Map on the right shows the locations in which the phrase *amendoeira em flor* occurs. Occ. is the number of occurrences for each phrase.

described in Section 3, replacing the GloVe word embeddings with the LX-DSemVectors[5] word embeddings created by Rodrigues et al. (2016) for the Portuguese language, and then combine the score S(p) of each candidate p with its Moran's I. If I(p) is the candidates Moran's I statistic. The final score FS(p) of a candidate is computed as follows:

$$\mathrm{FS}(p) = \mathrm{S}(p) + \frac{1 + \mathrm{I}(p)}{2} \tag{4.6}$$

The following text is one of the stories in the corpus, followed by the top 5 keyphrases extracted using the hybrid method, and the top 5 keyphrases weighed by their Moran's I as indicated in Equation 4.6.

O Diabo e as amêndoas

Conta-se que certo dia o Diabo, ao passar pelo termo de Uva, Vimioso, encontrou uma amendoeira em flor e sentou-se em baixo, pensando que estaria prestes a dar fruto. Entretanto passou por ali Nosso Senhor e perguntou-lhe: Que estás a fazer? Estou à espera

[5]https://github.com/nlx-group/lx-dsemvectors

que esta árvore dê fruto. Como já está em flor, não deve demorar. Não preferes antes ir esperar o fruto da cerejeira? Isso é que não. Esta já está em flor e a cerejeira ainda nem botões tem. E assim lá continuou sentado à espera que a amendoeira desse frutos. Entretanto passou o tempo, a cerejeira começou a florir e logo deu os seus frutos. E toda a gente se consolou com eles. Quanto à amendoeira, tudo estava muito atrasado. E o Diabo já começava a perder a paciência. Por fim, lá apareceram as amêndoas. O Diabo, cansado de esperar, tratou logo de as ir comendo. Só que por cada uma que comia era tão grande a carranca que fazia, que afugentava tudo à volta. Passados alguns dias voltou a comer e aconteceu a mesma coisa. As amêndoas continuavam amargas. Até que perdeu de vez a paciência e acabou por se ir embora, a rogar pragas, e sem ter comido uma única amêndoa de jeito.

Top 5 keyphrases extracted using the hybrid method:

amêndoa, cerejeira, fruto, amendoeira, fruto da cerejeira

Top 5 keyphrases extracted using the Moran's I statistic:

termo, amendoeira, tempo, amendoeira em flor, volta

Top 5 keyphrases extracted by combining the scores:

termo, amendoeira, amendoeira em flor, tempo, diabo

As we can see from the resulting keyphrases, the simple combination used in Equation 4.6 may not be the best one. The two new keyphrases that seem to be relevant to the text, *amendoeira em flor* and *diabo*, jumped to top 5 positions when weighed by their Moran's I, some other seemingly relevant terms, such as *fruto da cerejeira*, dropped out of the top 5. Furthermore, we can conclude that the combination given in Equation 4.6 is heavily biased towards the Moran's I value of the candidates, since the scores S(p) of every candidate keyphrase in a text sums to 1, while the Moran's I of every individual candidate is a value ranging between 1 and -1. To evaluate the quality of this combination we would need a corpus annotated with gold standard keyphrases and geospatial information about each text. These gold standard keyphrases would allow us to adjust the combination in Equation 4.6, by balancing the contributions of both the Moran's I statistic of a candidate p, and its score given by the hybrid method, in a way that increased the performance of the keyphrase extraction method.

4.5 Summary

This chapter presented the evaluation of the new keyphrase extraction method detailed in Chapter 3. It starts by presenting the methodology and evaluation metrics in Section 4.1, followed by a description, Section 4.2, of the corpora and how they were pre-processed in. Section 4.3 presents the evaluation results, along with a comparison with other state-of-the art keyphrase extraction methods, and a discussion of this comparison. Finally, Chapter 4.4 presents a case study in which a spatial autocorrelation metric, the Moran's I, is incorporated into the new method.

To summarize, the new keyphrase extraction method is effective at its purpose, having a good recall and precision at the SemEval2010 corpus. However, improvements to the method would be necessary in order to surpass other state-of-the art keyphrase extraction. The ascertain how the method could be improved, more testing would be needed, such as varying the parameters in all three corpus described in Section 4.2. Additionally, Section 4.4 shows that including spatial autocorrelation metrics can be useful in identifying keyphrases.

5 Conclusions and Future Work

5.1 Conclusions

This paper introduced a new hybrid method for keyphrase extraction, leveraging a combination of language models, graph-based ranking and word embedding vectors to estimate the importance of candidate keyphrases. The results obtained in the testing phase show that the proposed hybrid method is a viable approach to the keyphrase extraction task. The combination of different approaches used by the hybrid method approximates results obtained by state-of-the-art algorithms, and surpasses them in some cases. It was also demonstrated how spatial autocorrelation metrics could be incorporated into a keyphrase extraction method, and what reasoning exists for this inclusion. This Chapter overviews the main contributions of my M.Sc. thesis, as well a brief description of what experiments had their results discarded, and what future work could be considered for inclusion in this project.

5.2 Main Contributions

The main contributions of my M.Sc. thesis are the following:

- A novel keyphrase extraction method, combining a graph-based centrality measure, with a language model approach as a way to compute a prior probability distribution for candidate keyphrases. As well as a novel procedure for estimating the semantic relation between candidates, which is represented by the weight of the edge between two candidates in a graph representation of the text. This weight is a combination of a co-occurrence factor, with the distance between the word embedding vectors of each pair of candidates.
- A detailed evaluation of the new method on several corpora, to ascertain how the novel combination performs on different types of data (i.e., texts of different types and sizes). In addition to the evaluation, this thesis provides a comparison between the new method and state-of-the art related works, in an attempt to understand what differences between the methods could explain the differences in results.

- A case study in which a spatial distribution study was done on the lexicon of a corpus of Portuguese legends. This study allowed for the computation the Moran's of each candidate keyphrase in the corpus, and for the combination of the keyphrase extraction method with the Moran's I of each candidate. The results of this combination were analyzed, and a proposal was made on how to improve them by creating a corpus annotated with gold standard keyphrases and geographic information.

5.3 Future Work

The hybrid method proposed in this document could be tested on the new corpus for evaluating keyphrase extraction methods developed by Sterckx et al. (2017), in which, a set of 1467 articles from a dutch online publisher, with topics relating to fashion, sports, and general news, were manually annotated with keyphrases by 357 annotators.

Moreover, if we have a corpus with sufficient temporal information, having enough texts for different dates such that the distribution of occurrences for each date is not sparse, then we can extend the Moran's I as spatio-temporal autocorrelation measure. Gao (2015) presents the following variant of Equation 4.5 as a measure of spatio-temporal autocorrelation:

$$I_{st} = \frac{\sum_i^n \sum_j^n w_{ij}(\mathrm{p}_i(t) - \bar{\mathrm{p}}_t)(\mathrm{p}_j(t+\tau) - \bar{\mathrm{p}}_{t+\tau})}{\sigma_t \sigma_{t+\tau} \sum_i^n \sum_j^n w_{ij}} \tag{5.1}$$

In Equation 5.1, p is the target variable of interest, i and j are indices of total n spatial units, $\bar{\mathrm{p}}(t)$, $\bar{\mathrm{p}}_{t+\tau}$ are the means of variable p within a time lag, while σ_t and $\sigma_{t+\tau}$ are the variances. The local measures of spatio-temporal autocorrelation can be derived by decomposing a global measure into particular spatial neighboring units (Gao, 2015). By incorporating a temporal component in the Moran's I statistic, we can now attempt to identify candidates which have not only a particular geographic distribution, but also candidates that have a more unique temporal distribution (i.e., their occurrences tend to be clustered around neighbouring years).

The modification to the Knesser-Ney smoothing technique proposed by Shareghi et al. (2016) was also tested, and although it did not yield a higher performance, we cannot exclude the possibility that it would improve the results with a different combination of parameters. As such, further testing should be done with this improvement.

Another heuristic that was tested for inclusion in the new method was the usage of a dependency parser provided by Google's tensorflow, the idea was to limit potential candidate keyphrases to those where constituent words had a syntactic dependency to at least one other word inside the candidate. Similarly to the extension of the Knesser-Ney smoothing techniques, the results were marginally lower, as such, it would be interesting to revisit this idea with a different combination of parameters.

Finally, there are several other centrality measures that could be used as a replacement for the weighted PageRank approach. These measures were detailed in Section 2.2.1, and should be considered for testing.

Bibliography

Andor, D., Alberti, C., Weiss, D., Severyn, A., Presta, A., Ganchev, K., Petrov, S., and Collins, M. (2016). Globally Normalized Transition-Based Neural Networks. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Bougouin, A., Boudin, F., and Daille, B. (2013). TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction. In *Proceedings of the International Joint Conference on Natural Language Processing*.

Brill, E. (1995). Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-Of-Speech Tagging. *Computational linguistics*, 21(4).

Caragea, C., Bulgarov, F. A., Godea, A., and Gollapalli, S. D. (2014). Citation-Enhanced Keyphrase Extraction from Research Papers: A Supervised Approach. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Chen, S. F. and Goodmam, J. (1999). An Empirical Study of Smoothing Techniques for Language Modeling. *Computer Speech & Language*, 13(4).

Collins, M. (2002). Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural Language Processing (Almost) From Scratch. *Journal of Machine Learning Research*, 12(8).

Danesh, S., Sumner, T., and Martin, J. H. (2015). SGRank: Combining Statistical and Graphical Methods to Improve the State of the Art in Unsupervised Keyphrase Extraction. In *Proceedings of the Joint Conference on Lexical and Computational Semantics*.

El-Beltagy, S. R. and Rafea, A. (2009). KP-Miner: A Keyphrase Extraction System for English and Arabic Documents. *Information Systems*, 34(1).

Florescu, C. and Caragea, C. (2017). PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

Gao, S. (2015). Spatio-Temporal Analytics for Exploring Human Mobility Patterns and Urban Dynamics in the Mobile Age. *Spatial Cognition & Computation*, 15(2).

Ghahramani, Z. (2001). An Introduction to Hidden Markov Models and Bayesian Networks. *Journal of Pattern Recognition and Artificial Intelligence*, 15(01).

Grieve, J., Speelman, D., and Geeraerts, D. (2011). A Statistical Method for the Identification and Aggregation of Regional Linguistic Variation. *Language Variation and Change*, 23(02).

Hasan, K. S. and Ng, V. (2014). Automatic Keyphrase Extraction: A Survey of the State of the Art. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable Modified Kneser-Ney Language Model Estimation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics.*

Hulth, A. (2003). Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall.

Kim, S. N., Medelyan, O., Baldwin, T., and Kan, M.-Y. (2010). Automatic Keyphrase Extraction from Scientific Articles. In *Proceedings of the International Workshop on Semantic Evaluation.*

Kneser, R. and Ney, H. (1995). Improved Backing-off for M-gram Language Modeling. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing.*

Levy, O., Goldberg, Y., and Dagan, I. (2015). Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3(1).

Lopez, P. and Romary, L. (2010). HUMB: Automatic Key Term Extraction from Scientific Articles in GROBID. In *Proceedings of the International Workshop on Semantic Evaluation of the Association for Computational Linguistics.*

Mihalcea, R. and Tarau, P. (2004). TextRank: Bringing Order into Texts. In *Proceedings of the Conference on Empirical Methods on Natural Language.*

Mikolov, T., Yih, W.-t., and Zweig, G. (2013). Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics.*

Mitchell P. Marcus, M. A. M. and Santorini, B. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

Moran, P. (1950). Notes on Continuous Stochastic Phenomena. *Biometrika*, 37(1/2).

Nguyen, N. and Guo, Y. (2007). Comparisons of Sequence Labeling Algorithms and Extensions. In *Proceedings of the International Conference on Machine Learning.*

Over, P. and Yen, J. (2001). Introduction to DUC-2001: An Intrinsic Evaluation of Generic News Text Summarization Systems. In *Proceedings of the Document Understanding Workshop.*

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab.

Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.*

Petrov, S., Das, D., and McDonald, R. (2012). A Universal Part-of-Speech Tagset. In *Proceedings of the International Conference on Language Resources and Evaluation.*

Rodrigues, J., Branco, A., Neale, S., and Silva, J. (2016). LX-DSemVectors: Distributional Semantics Models for Portuguese. In *Proceedings of the International Conference on Computational Processing of the Portuguese Language.*

Rodrigues, R., Oliveira, H. G., and Gomes, P. (2014). LemPORT: a High-Accuracy Cross-Platform Lemmatizer for Portuguese. In *Proceedings of the Symposium on Languages, Applications and Technologies.*

Rousseau, F. and Vazirgiannis, M. (2015). Main Core Retention on Graph-of-Words for Single-Document Keyword Extraction. In *European Conference on Information Retrieval.*

Shareghi, E., Cohn, T., and Haffari, G. (2016). Richer Interpolative Smoothing Based on Modified Kneser-Ney Language Modeling. *Proceedings of the Conference on Empirical Methods in Natural Language Processing* .

Sterckx, L., Demeester, T., Deleu, J., and Develder, C. (2017). Creation and Evaluation of Large Keyphrase Extraction Collections With Multiple Opinions. *Language Resources and Evaluation*, 51(1).

Steven Bird, E. K. and Loper, E. (2009). *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit*. O'Reilly Media.

Sutton, C. and McCallum, A. (2012). An Introduction to Conditional Random Fields. *Foundations and Trends in Machine Learning*, 4(4).

Tomokiyo, T. and Hurst, M. (2004). A Language Model Approach to Keyphrase Extraction. In *Proceedings of the Association for Computational Linguistics Workshop on Multiword Expressions*.

Wan, X. and Xiao, J. (2008). Single Document Keyphrase Extraction Using Neighborhood Knowledge. In *Proceedings of the National Conference on Artificial Intelligence*.

Wang, R., Liu, W., and McDonald, C. (2014). Corpus-independent Generic Keyphrase Extraction Using Word Embedding Vectors. In *Proceedings of the Software Engineering Research Conference*.

Zhu, X., Goldberg, A. B., Van Gael, J., and Andrzejewski, D. (2007). Improving Diversity in Ranking Using Absorbing Random Walks. In *Proceeding of the Conference of the North American Chapter of the Association for Computational Linguistics*.