Keyphrase Extraction and Geospatial Characterizations for the Usage of Keyphrases

Francisco José Guerra Carreira

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 corpus 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.

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. 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).

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, they tend to be as computationally cheaper. Recent studies on NLP have proposed methods for unsupervised keyphrase extraction that combine different heuristics to model the relations between candidate terms.

My MSc 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., how related are the concepts described by different candidates). Additionally, the developed method also integrate 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.

Three corpus annotated with gold standard keyphrases were used to evaluate the performance of the the new method: 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. Results indicate that the new method can approximate and, in some cases, surpass, comparable state-of-the art keyphrase extraction methods. We also see that incorporating geospatial autocorrelation metrics can help in identifying keyphrases with a certain geographic uniqueness.

This paper is organized in the following way: Section 2 provides a summary of previous research related to unsupervised keyphrase extraction. Section 3 details the proposed method, enumerating the steps involved in candidate extraction, initial ranking, and candidate re-ranking. Section 4 describes the datasets used to evaluate the new hybrid method, and shows evaluation results regarding the performance of the method on those datasets. Section 5 details how geospatial information could be incorporated in keyphrase extraction. Finally, Section 6 presents the conclusion and ideas for future work.

2 Related Work

This section contains a description of existing work on unsupervised keyphrase extraction. Section 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 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.

2.1 Graph-Based Techniques for Keyphrase Extraction

There are several approaches that make use of a graph representation of the text as a way to estimate the importance of a candidate keyphrase (Hasan and Ng, 2014).

In the graph-based approach described by Mihalcea and Tarau (2004), the target text is first annotated using parts-of-speech (POS) tags and passed through a syntactic filter. This filter considers only nouns and adjectives. The lexical units that pass the filter 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 a centrality measure (Page et al., 1999), assuming that the importance of any candidate in a graph is defined by the number of edges connected to it, as well as the weight of those edges.

SGRank (Danesh et al., 2015) is a more recent 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 a subsumption count. For the position of first occurrence (PFO) factor, the authors propose using a logarithmic decay function computed as follows:

$$\mathrm{PFO}(t,d) = \log\left(\frac{cp}{\mathrm{p}(t,d)}\right) \tag{1}$$

In Equation 1, 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:

$$\mathrm{W_s}(t,d) = (\mathrm{TF}(t,d) - \mathrm{SC}(t,d)) \times \mathrm{IDF}(t) \times \mathrm{PFO}(t,d) \times \mathrm{TL}(t) \tag{2}$$

In Equation 2, $\mathrm{TL}(t)$ 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 weights 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, which is computed as follows:

$$\mathrm{W_d}(t_i,t_j) = \frac{\sum_{i=1}^{\mathrm{TF}(t_i)} \sum_{i=1}^{\mathrm{TF}(t_j)} \log\left(\frac{ws}{|pos_i - pos_j|}\right)}{\mathrm{NCO}(t_i,t_j)} \tag{3}$$

In Equation 3, $\mathrm{NCO}(t_i,t_j)$ 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 are included in the final edge weights $\mathrm{W_e}$, which are computed as follows:

$$\mathrm{W_e}(t_i,t_j) = \mathrm{W_d}(t_i,t_j) \times \mathrm{W_s}(t_i) \times \mathrm{W_s}(t_j) \tag{4}$$

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 the following equation, which corresponds to the iterative definition of the weighted PageRank method:

$$\mathrm{C_{WP}}(V_i) = \frac{(1-d)}{|V|} + d \times \sum_{V_j \in \mathrm{In}(V_i)} \frac{w_{ji}}{\sum_{V_k \in \mathrm{Out}(V_j)} w_{jk}} \times \mathrm{C_{WP}}(V_j) \tag{5}$$

Wang et al. (2014) proposed a weighing scheme that computes both phraseness and informativeness using information provided by word embedding vectors (Collobert et al., 2011) and local statistics. 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:

$$\mathrm{f}(w_i,w_j) = \frac{\mathrm{freq}(w_i) \times \mathrm{freq}(w_j)}{\mathrm{Euclidean}(w_i,w_j)^2} \tag{6}$$

In Equation 6, freq(w) is the frequency of the word w in d, and 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 6 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 6. Subsequently, Wang et al. (2014) compute the probability of two words co-occurring in a document d using the dice coefficient:

$$\text{Dice}(w_i, w_j) = \frac{2 \times \text{freq}(w_i, w_j)}{\text{freq}(w_i) + \text{freq}(w_j)} \tag{7}$$

In Equation 7, 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:

$$\text{Attr}(w_i, w_j) = \text{Dice}(w_i, w_j) \times \text{f}(w_i, w_j) \tag{8}$$

The previous weighing scheme can be used as a way to represent the weight on an edge connecting two candidates in a graph representation of the text. In Section 3.3 we will see a variation of Equation 6 that can be used to measure the semantic relation between two multi-word candidate keyphrases.

2.2 A Language Modeling Technique for Keyphrase Extraction

In addition to graph based methods there are also other approaches that use language models to estimate the importance of a candidate. One such method was proposed by Tomokiyo and Hurst (2004), they use a language model approach in which every sequence of words $W = w_1 w_2 w_3 \dots w_m$ 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 can be approximated as follows:

$$\begin{aligned}
\text{P}(W) &= \prod_{i=1}^{n} \text{P}(w_i \mid w_1, \dots, w_{i-1}) \\
&\approx \prod_{i=1}^{m} \text{P}(w_i \mid w_{i-(n-1)}, \dots, w_{i-1}) \\
&\approx \prod_{i=1}^{m} \frac{\#(w_{i-(n-1)}, \dots, w_{i-1}, w_i)}{\#(w_{i-(n-1)}, \dots, w_{i-1})} \\
&\approx \prod_{i=1}^{m} \frac{\#(w_{i-(n-1)}^{i-1}, w_i)}{\#(w_{i-(n-1)}^{i-1})}
\end{aligned} \tag{9}$$

In Equation 9, #() 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, in which, the probability of a sequence of words W is the product of the individual probabilities of each word w_i it contains. A unigram model is computed as follows:

$$\text{P}(W) \approx \prod_{i=1}^{m} \text{P}(w_i) \tag{10}$$

Having access to a background and a foreground corpus, and assuming that we want to extract keyphrases from a foreground corpus whose documents are similar to those in the background 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 a phrase) and informativeness (i.e., how well the phrase captures the key ideas in the foreground corpus). We denote the different language models that are involved as: LM^1_{fg}

for a foreground unigram model; LM^1_{bg} for a background unigram model; LM^N_{fg} and LM^N_{bg} for foreground and background N-th order language models, respectively.

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

$$\delta_W(\text{p} \parallel \text{q}) = \text{p}(W) \times \log\left(\frac{\text{p}(W)}{\text{q}(W)}\right) \tag{11}$$

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

$$\delta_W(LM^N_{\text{fg}} \parallel LM^1_{\text{fg}}) \tag{12}$$

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:

$$\delta_W(LM^N_{\text{fg}} \parallel LM^N_{\text{bg}}) \tag{13}$$

$$\delta_W(LM^1_{\text{fg}} \parallel LM^1_{\text{bg}}) \tag{14}$$

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. In the hybrid method described in Section 3, the smoothing method that was used is the widely adopted method proposed by Chen and Goodmam (1999), which is an extension to the number of discounts used in the Kneser-Ney smoothing technique (Kneser and Ney, 1995). The reader is referred to the article by Zhai (2008) for an in depth introduction to language model estimation and parameter smoothing.

3 A Hybrid Method for Keyphrase Extraction

This section 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 model, 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.

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

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.

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 and further filter candidate keyphrases. The threshold is a parameter that can be adjusted if, as an example, a corpus is composed of smaller texts, for which this threshold would allow most candidates to be included in the graph. Each vertex p is given a prior weight computed as the weighted sum of the phraseness and informativeness score of each candidate, using Equations 12 and 13 respectively. The probabilities for each candidates, needed to compute the phraseness and informativeness of each candidate, are given by the language models described previously. Assuming that wf 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 their prior, then the prior weight $\text{Prior}(p)$ of a candidate p is computed as follows:

$$\text{Prior}(p) = wf \times \text{Phraseness}(p) + (1 - wf) \times \text{Informativeness}(p) \tag{15}$$

In Equation 15, the wf 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 $wf = 1$, we can consider only the informativeness of a candidate by setting $wf = 0$, 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:

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

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

$$\text{MPrior}(p) = \text{Prior}(p) \times \left(\frac{1}{21 + line}\right) \times \left(\frac{1}{e^{|(2-|p|)|}}\right) \tag{16}$$

In Equation 16, *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 15.

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. 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:

$$\mathbf{V}_p = \frac{\sum_{i=1}^{n} \text{idfs}(w_i) \times \mathbf{e}_i}{|p|} \tag{17}$$

In Equation 17, $|p|$ is the number of words in p, and 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 dfs(w_i) be the number of documents in which w_i occurs, then idfs(w_i) is computed as follows:

$$\text{idfs}(w_i) = 50 + \log\left(\frac{|Corpus|}{\text{dfs}(w_i)}\right) \tag{18}$$

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 similarity tasks. A measure of similarity 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 Contains(p_i, p_j) be a function that is true if p_i is contained in p_j, and let 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:

$$\text{Sim}(p_i, p_j, S) = \begin{cases} \text{MPrior}(p_i) \times \text{MPrior}(p_j) & \text{if Contains}(p_i, p_j) \\ \frac{\text{MPrior}(p_i) \times \text{MPrior}(p_j)}{e^{(\text{Euclidean}(\mathbf{F}_{p_i}, \mathbf{F}_{p_j}))}} \times \frac{1}{\log(2+\text{Distance}(p_i, p_j, S))} & \text{otherwise} \end{cases} \tag{19}$$

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

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:

$$\text{Weight}(p_i, p_j) = \sum_{z=1}^{n} \text{Sim}(p_i, p_j, S_z)^{\beta} \tag{20}$$

In Equation 20, β is a parameter whose value is 1 if $|p_i| \leq |p_j|$ and 2 otherwise. Finally, we compute the score $\text{S}(p)$ 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:

$$\text{S}(p_i) = (1-d) \times \frac{\text{MPrior}(p_i)}{\sum_{p_j} \text{MPrior}(p_j)} + d \times \sum_{p_j \in \text{Links}(p_i)} \frac{\text{S}(p_j) \times \text{Weight}(p_i, p_j)}{\sum_{p_k \in \text{Links}(p_j)} \text{Weight}(p_j, p_k)} \tag{21}$$

In Equation 21, the parameter d is set to 0.4, and the functions $\text{MPrior}(p)$ and $\text{Weight}(p_i, p_j)$, are computed thorough Equations 16 and 20, 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.

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 wf 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 section.

4 Experimental evaluation

The evaluation of the hybrid method was done by measuring the performance on different corpora[4] commonly used in evaluating keyphrase extraction algorithms. A summary of the number of documents and keyphrases in each corpus is present in Table 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

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

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

Table 1: Number of documents, keyphrases by corpus, and average number of keyphrases per document.

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 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.

Performance was measured using the Precision, Recall and F1 metrics, and compared against 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[5] Python library. Performance was measured on the top 5, 10, and 15 highest ranked candidates by each algorithm. These candidates are stemmed before being matched to the gold standard keyphrases of each corpus. 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. 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 21, 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 prior weights).

Table 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 15 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 3 the original results presented by the authors on the corpora described in this section. Some data in Table 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 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

[5]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	0.234	0.194	0.167
	R	0.081	0.106	0.124	0.091	0.137	0.181	0.079	0.092	0.100
	F	0.097	0.107	0.111	0.112	0.141	0.162	0.118	0.125	0.125
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 2: Performance of different methods at the 5, 10, and 15 top ranked candidates.

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

several noteworthy observations from Table 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 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 15), 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 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 3, may not be the highest scores the Complete method could attain in both the Inspec and DUC2001 corpora.

5 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 metric 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 $\mathbf{W} = \{w\}_{i,j \in \{1...n\}}$ 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:

$$w_{ij} = \begin{cases} 1, & d_{ij} < \tau,\ i \neq j \\ 0, & d_{ij} \geq \tau,\ \text{or } i = j \end{cases} \tag{22}$$

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 $\mathbf{OP} = \{o_{p_1}, ..., o_{p_i}\}$, 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:

$$\mathrm{I}(p) = \frac{n}{\sum_i^n (o_{p_i} - \bar{o}_p)} \times \frac{\sum_i^n \sum_j^n w_{ij}(o_{p_i} - \bar{o}_p)(o_{p_j} - \bar{o}_p)}{\sum_i^n \sum_j^n w_{ij}} \tag{23}$$

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 [6] 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 23, 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[7] 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 23.

Figure 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 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 described in Section 3, replacing the GloVe word embeddings with the LX-DSemVectors[8] word embeddings created

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

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

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

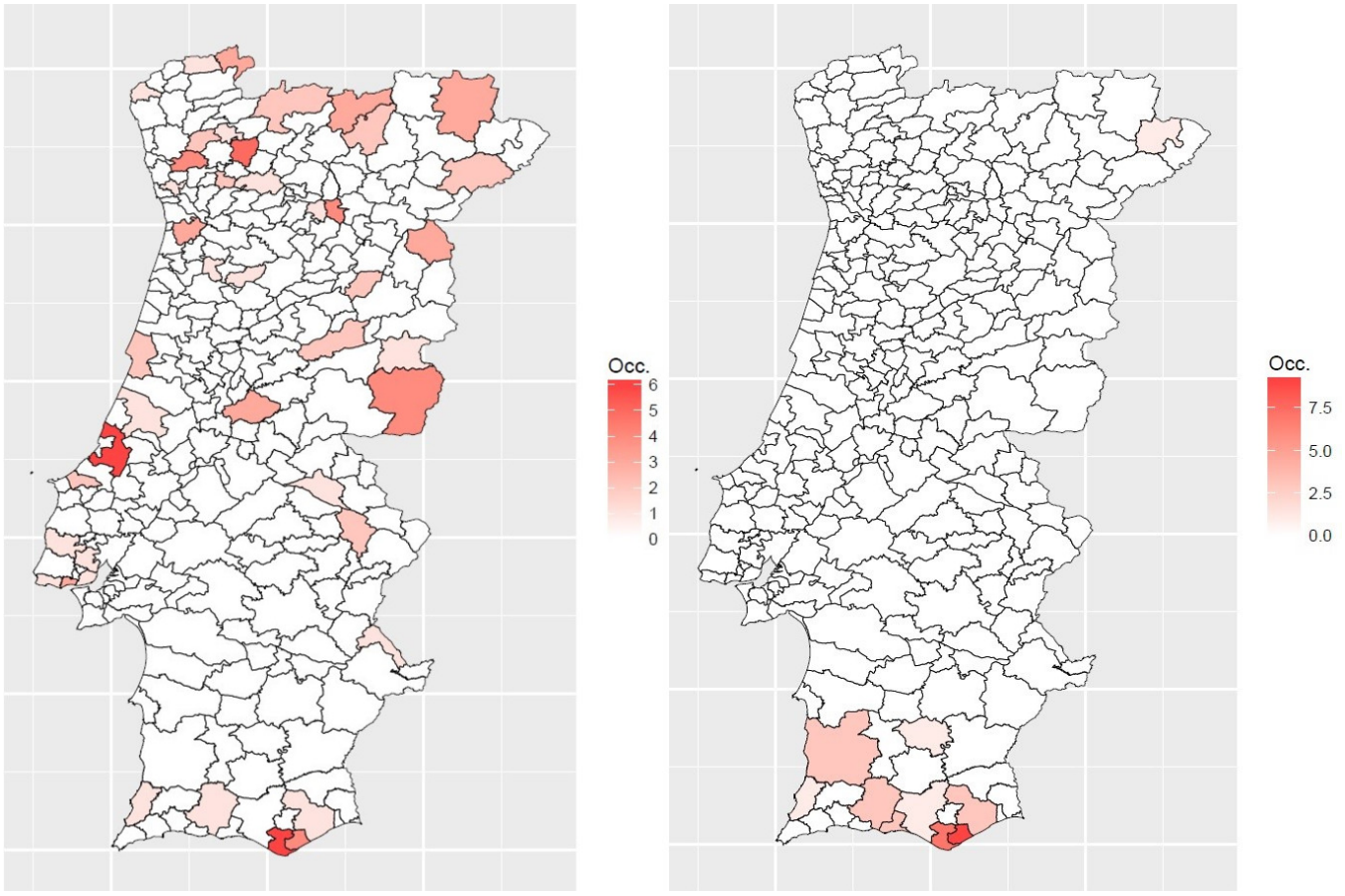

Figure 1: Map on the left shows the locations in which the phrase *santa maria* occurs. Map on the right shows the locations in which the phrase *amendoeira em flor* occurs. Occ. is the number of occurrences for each phrase.

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 p Moran's I statistic. The final score $FS(p)$ of a candidate is computed as follows:

$$FS(p) = S(p) + \frac{1 + I(p)}{2} \tag{24}$$

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 24.

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 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 Moran's I statistic with the hybrid method:
termo, amendoeira, amendoeira em flor, tempo, diabo

As we can see from the resulting keyphrases, the simple combination used in Equation 24 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 24 is heavily biased towards the Moran's I value of the candidates, since the scores $\mathrm{S}(p)$ 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 24, 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.

6 Conclusions and Future Work

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 not only approximates results obtained by state-of-the-art algorithms, but surpasses them in some cases. It is worth noting that the tuning of hyper-parameters, such as the number of vertices in the graph, can be used to improve the performance on different corpora. We can see that in a corpus such as Inspec, where the size of each document is small in comparison to the SemEval or DUC2001 corpora, the top 900 candidates of each document will represent a higher proportion of the total number of candidates. As such, this is not such a particularly restrictive filter in that particular corpus.

Furthermore, the hybrid method 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 23 as a measure of spatio-temporal autocorrelation:

$$I_{st} = \frac{\sum_i^n \sum_j^n w_{ij}(\mathrm{p_i}(t) - \bar{\mathrm{p}}_t)(\mathrm{p}_j(t+\tau) - \bar{\mathrm{p}}_{t+\tau})}{\sigma_t \sigma_{t+\tau} \sum_i^n \sum_j^n w_{ij}} \tag{25}$$

In Equation 25, 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).

Finally, Shareghi et al. (2016) proposed a modification to the Knesser-Ney smoothing technique by adding additional discount values. 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. Because we use Knesser-Ney smoothing when estimating the language models from which we compute the prior probabilities for every candidate, this new extensions could improve the method proposed in this paper. We did a modification to the KenLM toolkit to accommodate the extra parameters, but, the results were marginally lower and were discarded. However, 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.

References

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*.

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

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).

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

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*.

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*.

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

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*.

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).

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*.

Zhai, C. (2008). Statistical Language Models for Information Retrieval. *Synthesis Lectures on Human Language Technologies*, 1(1).