Reducing Over-generation Errors for Automatic Keyphrase Extraction using Integer Linear Programming
Florian Boudin

To cite this version:
Florian Boudin. Reducing Over-generation Errors for Automatic Keyphrase Extraction using Integer Linear Programming. ACL 2015 Workshop on Novel Computational Approaches to Keyphrase Extraction, Jul 2015, Pékin, China. hal-01185669

HAL Id: hal-01185669
https://hal.science/hal-01185669
Submitted on 21 Aug 2015
Reducing Over-generation Errors for Automatic Keyphrase Extraction using Integer Linear Programming

Florian Boudin
LINA - UMR CNRS 6241, Université de Nantes, France
florian.boudin@univ-nantes.fr

Abstract

We introduce a global inference model for keyphrase extraction that reduces over-generation errors by weighting sets of keyphrase candidates according to their component words. Our model can be applied on top of any supervised or unsupervised word weighting function. Experimental results show a substantial improvement over commonly used word-based ranking approaches.

1 Introduction

Keyphrases are words or phrases that capture the main topics discussed in a document. Automatically extracted keyphrases have been found to be useful for many natural language processing and information retrieval tasks, such as summarization (Litvak and Last, 2008), opinion mining (Berend, 2011) or text categorization (Hulth and Megyesi, 2006). Despite considerable research effort, the automatic extraction of keyphrases that match those of human experts remains challenging (Kim et al., 2010).

Recent work has shown that most errors made by state-of-the-art keyphrase extraction systems are due to over-generation (Hasan and Ng, 2014). Over-generation errors occur when a system correctly outputs a keyphrase because it contains an important word, but at the same time erroneously predicts other keyphrase candidates as keyphrases because they contain the same word. One reason these errors are frequent is that many unsupervised systems rank candidates according to the weights of their component words, e.g. (Wan and Xiao, 2008a; Liu et al., 2009), and many supervised systems use unigrams as features, e.g. (Turney, 2000; Nguyen and Luong, 2010).

While weighting words instead of phrases may seem rather blunt, it offers several advantages. In practice, words are usually much easier to extract, match and weight, especially for short documents where many phrases may not be statistically frequent (Liu et al., 2011).

Selecting keyphrase candidates according to their component words may also turn out to be useful for reducing over-generation errors if one can ensure that the importance of each word is counted only once in the set of extracted keyphrases. To do so, keyphrases should be extracted as a set rather than independently. Finding the optimal set of keyphrases is a combinatorial optimization problem, and can be formulated as an integer linear program (ILP) which can be solved exactly using off-the-shelf solvers.

In this work, we propose an ILP formulation for keyphrase extraction that can be applied on top of any word weighting scheme. Through experiments carried out on the SemEval dataset (Kim et al., 2010), we show that our model increases the performance of both supervised and unsupervised word weighting keyphrase extraction methods.

The rest of this paper is organized as follows. In Section 2, we describe our ILP model for keyphrase extraction. Our experiments are presented in Section 3. In Section 4, we briefly review the previous work, and we conclude in Section 5.

2 Method

Our global inference model for keyphrase extraction consists of three steps. First, keyphrase candidates are extracted from the document using heuristic rules. Second, words are weighted using either supervised or unsupervised methods. Third, finding the optimal subset of keyphrase candidates is cast as an ILP and solved using an off-the-shelf solver.

2.1 Keyphrase candidate selection

Candidate selection is the task of identifying the words or phrases that have properties similar to
those of manually assigned keyphrases. First, we apply the following pre-processing steps to the document: sentence segmentation\(^1\), word tokenization\(^2\) and Part-Of-Speech (POS) tagging\(^3\).

Following previous work (Wan and Xiao, 2008a; Bougouin et al., 2013), we use the sequences of nouns and adjectives as keyphrase candidates. Candidates that have less than three characters, that contain only adjectives, or that contain stop-words\(^4\) are filtered out. These heuristic rules are designed to avoid spurious instances and keep the number of candidates to a minimum (Hasan and Ng, 2014). All words are stemmed using Porter’s stemmer (Porter, 1980).

2.2 Word weighting functions

The performance of our model depends on how word weights are estimated. Here, we experiment with three methods for assigning importance weights to words. The first two are unsupervised weighting functions, namely \(\text{TF} \times \text{IDF}\) (Spärck Jones, 1972) and TextRank (Mihalcea and Tarau, 2004), which have been extensively used in prior work (Hasan and Ng, 2010). We also apply a supervised model for predicting word importance based on (Hong and Nenkova, 2014).

2.2.1 TF×IDF

The weight of each word \(t\) is estimated using its frequency \(tf(t,d)\) in the document \(d\) and how many other documents include \(t\) (inverse document frequency), and is defined as:

\[
\text{TF} \times \text{IDF}(t,d) = tf(t,d) \times \log(D/D_t)
\]

where \(D\) is the total number of documents and \(D_t\) is the number of documents containing \(t\).

2.2.2 TextRank

A co-occurrence graph is first built from the document in which nodes are words and edges represent the number of times two words co-occur in the same sentence. TextRank (Mihalcea and Tarau, 2004), a graph-based ranking algorithm, is then used to compute the importance weight of each word. Let \(d\) be a damping factor\(^5\), the TextRank score \(S(V_i)\) of a node \(V_i\) is initialized to a default value and computed iteratively until convergence using the following equation:

\[
S(V_i) = (1 - d) + \left( d \times \sum_{V_j \in \mathcal{N}(V_i)} \frac{w_{ji} \times S(V_j)}{\sum_{V_k \in \mathcal{N}(V_j)} w_{jk}} \right)
\]

where \(\mathcal{N}(V_i)\) is the set of nodes connected to \(V_i\) and \(w_{ji}\) is the weight of the edge between nodes \(V_j\) and \(V_i\).

TextRank implements the concept of “voting”, i.e. a word is important if it is highly connected to other words and if it is connected to important words.

2.2.3 Logistic regression

We train a logistic regression model\(^6\) for assigning importance weights to words in the document based on (Hong and Nenkova, 2014). Reference keyphrases in the training data are used to generate positive and negative examples. For a word in the document (restricted to adjectives and nouns), we assign label 1 if the word appears in the corresponding reference keyphrases, otherwise we assign 0. We use the relative position of the first occurrence, the presence in the first sentence and the TF×IDF weight as features. These features have been extensively used in supervised keyphrase extraction approaches, and have been shown to perform consistently well (Hasan and Ng, 2014).

2.3 ILP model definition

Our model is an adaptation of the concept-based ILP model for summarization introduced by (Gillick and Favre, 2009), in which sentence selection is cast as an instance of the budgeted maximum coverage problem\(^7\). The key assumption of our model is that the value of a set of keyphrase candidates is defined as the sum of the weights of the unique words it contains. That way, a set of candidates only benefits from including each word once. Words are thus assumed to be independent, that is, the value of including a word is not affected by the presence of any other word in the set of keyphrases.

Formally, let \(w_i\), \(x_i\) and \(c_j\) two binary variables indicating the presence of a word in a set, and a positive weight assigned to it.

\(^1\)We use Punkt Sentence Tokenizer from NLTK.
\(^2\)We use Penn Treebank Tokenizer from NLTK.
\(^3\)We use the Stanford Part-Of-Speech Tagger (Toutanova et al., 2003).
\(^4\)We use the english stop-list from NLTK.
\(^5\)We set \(d\) to 0.85 as in (Mihalcea and Tarau, 2004).
\(^6\)We use the Logistic Regression classifier from scikit-learn with default parameters.
\(^7\)Given a collection \(S\) of sets with associated costs and a budget \(L\), find a subset \(S' \subseteq S\) such that the total cost of sets in \(S'\) does not exceed \(L\), and the total weight of elements covered by \(S'\) is maximized (Kuller et al., 1999).
ence of word $i$ and candidate $j$ in the set of extracted keyphrases, $\text{Occ}_{ij}$ an indicator of the occurrence of word $i$ in candidate $j$ and $N$ the maximum number of extracted keyphrases, our model is described as:

$$\text{max} \sum_i w_i x_i \quad (1)$$

$$s.t. \sum_j c_j \leq N \quad (2)$$

$$c_j \text{Occ}_{ij} \leq x_i, \quad \forall i, j \quad (3)$$

$$\sum_j c_j \text{Occ}_{ij} \geq x_i, \quad \forall i \quad (4)$$

$$x_i \in \{0, 1\} \quad \forall i$$

$$c_j \in \{0, 1\} \quad \forall j$$

The constraints formalized in equations 3 and 4 ensure the consistency of the solution: selecting a candidate leads to the selection of all the words it contains, and selecting a word is only possible if it is present in at least one selected candidate.

By summing over word weights, this model overly favors long candidates. Indeed, given two keyphrase candidates, one being included in the other (e.g. uddi registries and multiple uddi registries), this model always selects the longest one as its contribution to the objective function is larger. To correct this bias, a regularization term is added to the objective function:

$$\text{max} \sum_i w_i x_i - \lambda \sum_j \frac{(l_j - 1)c_j}{1 + \text{substr}_j} \quad (5)$$

where $l_j$ is the size, in words, of candidate $j$, and $\text{substr}_j$ the number of times $c_j$ occurs as a substring in the other candidates. This regularization penalizes the candidates that are composed of more than two words, and is dampened for candidates that occur frequently as substrings in other candidates. Here, we assume that for multiple candidates of the same size, the one that is less frequent in the document should be stressed first.

The resulting ILP is then solved exactly using an off-the-shelf solver\(^3\). The solving process takes less than a second per document on average. The $N$ candidate keyphrases returned by the solver are selected as keyphrases.

\(^3\)We use GLPK, http://www.gnu.org/software/glpk/

3 Experiments

3.1 Experimental settings

We carry out our experiments on the SemEval dataset (Kim et al., 2010), which is composed of scientific articles collected from the ACM Digital Library. The dataset is divided into training (144 documents) and test (100 documents) sets. We use the set of combined author- and reader-assigned keyphrases as reference keyphrases.

We follow the common practice (Kim et al., 2010) and evaluate the performance of our method in terms of precision ($P$), recall ($R$) and $f$-measure ($F$) at the top $N$ keyphrases\(^9\). Extracted and reference keyphrases are stemmed to reduce the number of mismatches.

For each word weighting function, namely TF×IDF, TextRank and Logistic regression, we compare the performance of our ILP model (hereafter \textit{ilp}) with that of two word-based weighting baselines. The first baseline (hereafter \textit{sum}) simply ranks keyphrase candidates according to the sum of the weights of their component words as in (Wan and Xiao, 2008b; Wan and Xiao, 2008a). The second baseline (hereafter \textit{norm}) consists in scoring keyphrase candidates by computing the sum of the weights of their component words normalized by their length as in (Boudin, 2013).

As a post-processing step, we remove redundant keyphrases from the ranked lists generated by both baselines. A keyphrase is considered redundant if it is included in another keyphrase that is ranked higher in the list.

IDF weights are computed on the training set. The regularization parameter $\lambda$ is set, for all the experiments, to the value that achieves the best performance on the training set, that is 0.3 for TF×IDF, 0.4 for TextRank and 1.2 for Logistic regression.

3.2 Results

The performance of our model on top of different word weighting functions is shown in Table 1. Overall, our model consistently improves the performance over the baselines. We observe that the results for \textit{sum} are very low. Summing the word weights favors long candidates and is prone to over-generation errors, as illustrated by the example in Table 2.

\(^9\)Scores are computed using the evaluation script provided by the SemEval organizers.
Table 1: Comparison of TF×IDF, TextRank and Logistic regression for different ranking strategies when extracting a maximum of 5 and 10 keyphrases. Results are expressed as a percentage of precision (P), recall (R) and f-measure (F). † indicates significance at the 0.05 level using Student’s t-test.

<table>
<thead>
<tr>
<th>Weighting + Ranking</th>
<th>Top-5 candidates</th>
<th>Top-10 candidates</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF×IDF + sum</td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>+ norm</td>
<td>19.2</td>
<td>6.7</td>
</tr>
<tr>
<td>+ ilp</td>
<td>25.4</td>
<td>9.1</td>
</tr>
<tr>
<td>TextRank + sum</td>
<td>4.5</td>
<td>1.6</td>
</tr>
<tr>
<td>+ norm</td>
<td>18.8</td>
<td>6.6</td>
</tr>
<tr>
<td>+ ilp</td>
<td>22.6</td>
<td>8.0</td>
</tr>
<tr>
<td>Logistic regression</td>
<td>4.2</td>
<td>1.5</td>
</tr>
<tr>
<td>+ norm</td>
<td>23.8</td>
<td>8.3</td>
</tr>
<tr>
<td>+ ilp</td>
<td>29.4</td>
<td>10.4</td>
</tr>
</tbody>
</table>

Normalizing the candidate scores by their lengths (norm) produces shorter candidates but does not limit the number of over-generation errors. As we can see from the example in Table 2, 9 out of 10 extracted keyphrases are containing the word nugget. Our ILP model removes these redundant keyphrases by controlling the impact of each word on the set of extracted keyphrases. The resulting set of keyphrases is more diverse and thus increases the coverage of the topics addressed in the document.

Note that the reported results are not on par with keyphrase extraction systems that use ad-hoc pre-processing, involve structural features and leverage external resources. Rather our goal in this work is to demonstrate a simple and intuitive model for reducing over-generation errors.

4 Related Work

In recent years, keyphrase extraction has attracted considerable attention and many different approaches were proposed. Generally speaking, keyphrase extraction methods can be divided into two main categories: supervised and unsupervised approaches.

Supervised approaches treat keyphrase extraction as a binary classification task, where each phrase is labeled as keyphrase or non-keyphrase (Witten et al., 1999; Turney, 2000; Kim and Kan, 2009; Lopez and Romary, 2010). Unsupervised approaches usually rank phrases by importance and select the top-ranked ones as keyphrases. Methods for ranking phrases include graph-based ranking (Mihalcea and Tarau, 2004; Wan and Xiao, 2008a; Wan and Xiao, 2008b; Bougouin et al., 2013; Boudin, 2013), topic-based clustering (Liu et al., 2009; Liu et al., 2010; Bougouin et al., 2013), statistical models (Paukkeri and Honkela, 2010; El-Beltagy and Rafea, 2010) and language modeling (Tomokiyo and Hurst, 2003).

The work of (Ding et al., 2011) is perhaps the closest to our present work. They proposed an ILP formulation of the keyphrase extraction prob-
lem that combines TF×IDF and position features in an objective function subject to constraints of coherence and coverage. In their model, coherence is measured by Mutual Information and coverage is estimated using Latent Dirichlet Allocation (LDA) (Blei et al., 2003). Their work differs from ours in that (1) it is phrased-based and thus does not penalize redundant keyphrases, and (2) it requires estimating a large number of hyperparameters which makes it difficult to generalize.

5 Conclusion and Future Work

In this paper, we proposed an ILP formulation for keyphrase extraction that reduces over-generation errors by weighting keyphrase candidates as a set rather than independently. In our model, keyphrases are selected according to their component words, and the weight of each unique word is counted only once. Experiments show a substantial improvement over commonly used word-based ranking approaches using either supervised and unsupervised weighting schemes.

In future work, we intend to extend our model to include word relatedness through the use of association measures. By doing so, we expect to better differentiate semantically related keyphrase candidates according to the association strength between their component words.

Acknowledgments

This work was partially supported by the GOLEM project (grant of CNRS PEPS FaSciDo 2015, http://boudinfl.github.io/GOLEM/). We thank the anonymous reviewers, Adrien Bougouin and Evgeny Gurevsky for their insightful comments.

References


