A knowledge-base oriented approach for automatic keyword extraction

Ludovic Jean-Louis
Ecole Poytechnique de Montréal
ludovic.jean-louis@polymtl.ca

Michel Gagnon
Ecole Poytechnique de Montréal
michel.gagnon@polymtl.ca

Eric Charton
Centre de Recherche Informatique de Montréal
eric.charton@crim.ca

ABSTRACT
Automatic keyword extraction is an important subfield of information extraction process. It is a difficult task, where numerous different techniques and resources have been proposed. In this paper, we propose a generic approach to extract keyword from documents using encyclopedic knowledge. Our two-step approach first relies on a classification step for identifying candidate keywords followed by a learning-to-rank method depending on a user-defined keyword profile to order the candidates. The novelty of our approach relies on i) the usage of the keyword profile ii) generic features derived from Wikipedia categories and not necessarily related to the document content. We evaluate our system on keyword datasets and corpora from standard evaluation campaign and show that our system improves the global process of keyword extraction.

1. INTRODUCTION
Small sets of keywords are commonly used as a way to summarize the content of a document. Since identifying keywords for a document requires domain knowledge, it is in some cases mandatory that the writers provide a set of keywords together with their documents. For instance, technical book writers or researchers must submit a set of keywords together with their publication. The task of assigning keywords is also useful in other domains where the writers usually do not provide them (e.g in emails or web pages documents). To tackle this issue, various methods have been proposed to automatically extract a set of keywords based on a document. In this paper, we propose an automatic keyword extraction method dedicated to the identification of similar documents in a collection. This work have been conducted in the context of a global system for email categorization. This system proceed in three steps : i) extraction of keywords in a given document; ii) expansion of the initial set of keywords by finding related terms; iii) use of the keyword expansions to search for related documents. This paper focuses only on the first step, keyword extraction.

To extract the appropriate keywords from a document, we first adopt a classical machine learning-based approach. We improve it with features based on general knowledge derived from Wikipedia. Wikipedia categories have a large coverage of different domains, which makes it interesting for extracting keywords on large data set. Our idea is to use semantic information provided by Wikipedia categories to improve keywords identification. Once candidate keywords have been extracted, they are ranked according to a learning-to-rank approach. The ranking step is based on a user-defined keyword profile that specifies the user preferences in terms of domain knowledge. This profile improves the accuracy of extracted keywords.

This paper is structured as follows. In section 2, we present various approaches related to keyword extraction. We then describe the proposed method for candidate keyword extraction in Section 3.1 and the ranking method in Section 3.2. Our approach is tested on different test samples and the results obtained are presented and commented in Section 4. We conclude in Section 5.

2. RELATED WORK
An accurate set of keywords can be used in a wide range of information extraction tasks such as document indexing and retrieval, categorization, or filtering [1]. Numerous algorithms have been proposed for extraction of keywords and experimented with different types of documents. Textual documents (news articles, scientific publications, web pages) can be categorized with keywords in a classification process [2, 3], or exploited in a more complex application, like content-based advertising [4, 5, 6]. Two main approaches can be distinguished for keyword extraction: centered on the document content or driven according to external knowledge.

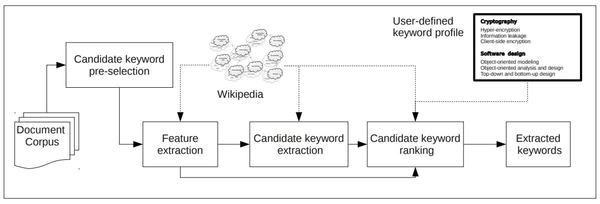
2.1 Document centric keyword extraction
Machine learning-based methods are the most used in document-centric keyword extraction. They consider keyword extraction as a classification problem where a model is utilized to determine whether a term (or an n-gram) is a valid keyword [7, 8, 9, 10]. Frequency-based is another approach which goal is to index terms in a document or a corpus according to a measure of their weights or frequencies. Frequency-based methods are unsupervised approaches that mainly rely on the frequency distribution of the terms in a single document [11, 12]. These are domain-independent methods, that do not take into account the structure of the document. For example, in the *KEA* system (Keyphrase Extraction Algorithm)[7], candidate keywords are generated as a set of n-grams derived from the document according to their delimiters, like punctuation marks and then classified by a Naive Bayes model. Features used are TF-IDF weight and position of the first occurrence of the candidate in the document. Various different document-centered features have been suggested; they use patterns of part-of-speech tags [9],

spread (distance between the first and the last occurrence) [13], position of the candidate in different parts of the document, etc. Other approaches have tackled the problem as a sequence labeling task. Zhang [10] proposed to use a CRF for extracting keywords from a document, where the CRF model is used to identify the boundaries of the candidate keywords.

2.2 Keyword extraction using external knowledge

Another family of keyword extraction systems exploits knowledge from an external source to improve the process. Maui [13] relies on semantic knowledge from Wikipedia. Semantic information is used to derive several Wikipedia-based features during candidate keyword identification: i) *Wikipedia keyphraseness:* likelihood of a term being a link in Wikipedia; ii) *semantic relatedness:* semantic score derived from Wikipedia associated with each candidate keyword[1]; iii) *inverse Wikipedia linkage*, which is the normalized number of pages that link to a Wikipedia page. Another example of approach based on Wikipedia is presented in [14]. A graph is built by using the candidate keywords and the semantic relatedness of these terms in Wikipedia. The idea is that most interesting keywords tend to concentrate into sub-graphs. To localize keywords, a network analysis algorithm (Girvan-Newman) is applied to identify densely interconnected subgraphs and extract keywords from them. More recently, [6] proposed different strategies that also rely on Wikipedia data. The keywords are identified based on their number of occurrences in Wikipedia titles and the categories associated with these titles. It is also interesting to note that in the past years a shared task on keyword extraction has been proposed in the SemEval evaluation [15]. The results of that evaluation have shown that systems that take into account information from external sources achieve better performance than others.

Figure 1: Overview of our keyword extraction approach


User-defined keyword profile
Cryptography
Hyper-encryption
Information leakage
Client-side encryption
Software design
Object-oriented modeling
Object-oriented analysis and design
Top-down and bottom-up design
Candidate keyword pre-selection
Wikipedia
Document Corpus
Feature extraction
Candidate keyword extraction
Candidate keyword ranking
Extracted keywords

3. DESCRIPTION OF OUR METHOD FOR KEYWORD EXTRACTION

In this paper, we present a generic approach, using machine learning methods. Our system first extracts candidate keywords from a document and computes a set of features from these candidates in order to evaluate their likelihood of being relevant keywords. Then the resulting set is ordered according to a confidence score that represents the relevancy of each keyword. For both tasks, semantic information from Wikipedia is involved: i) we use the Wikipedia categories and their instances for generating features for candidate keyword extraction; ii) we compute a ranking score for the candidate keywords founded on a keyword profile. In short, the keyword profile is a set of terms (provided by a user) and categories associated with these terms that are extracted automatically from Wikipedia. Figure 1 gives an overview of our approach.

[1]This feature is calculated by linking each candidate keyword to a Wikipedia article, then comparing – by means of a Wikipedia-based semantic measure a given candidate keyword to all the other candidates. The final value corresponds to the total of all the pairwise comparisons.

3.1 Method for candidate keyword extraction

The candidate keyword extraction process is conducted in two steps: first we generate candidate keywords based on the n-grams contained in a document, then a classifier is applied to each candidate keyword to decide whether it is actually a keyword.

3.1.1 Selection of the candidate keywords

The following rules are used to discard word-sequence that cannot be considered as keywords:

- Split the document according to punctuation marks and stop words;
- generate all n-grams between two stop-words;
- remove candidates containing special characters (non-alphabetic symbols);
- remove candidates having less than l characters;
- remove candidates having more than t tokens;
- remove candidates having more than h hyphens;
- remove candidates having more than d digits[2];
- remove candidates starting/ending with an adjective or an adverb;
- remove candidates having a token that belongs to a black list;
- remove candidates having duplicate tokens;
- validate the candidate keyword against a list of accepted POS tag patterns.

3.1.2 Features for candidate keywords extraction

The feature set includes features that have proven to be efficient in previous work as well as features extracted from external sources. Two external sources are used: Wikipedia categories and a large corpus, described in Section 4. The external corpus approximates frequency information that we would obtain by querying an online search engine. The features used by the classifier are defined as follows:

d – a document in the corpus,

t – a candidate keyword in d,

$|t|$ – number of tokens in t,

w – a word in d,

$f(t, d)$ – number of occurrences of t in d,

$|D|$ – number of documents in the corpus,

$|K|$ – number of documents in the external domain independent corpus,

$df(t, D)$ – number of documents in D where t is mentioned ($|\{d \in D, t \in d\}|$),

[2]$l = 3$ characters, $t = 8$ tokens, $h = 3$ hyphens, $d = 1$ digit.

$len(t)$ – function that returns the number of characters in t,

$firstLine$ – first line of the document d,

$rightIndex(t, d)$ – offset of the first occurrence of t in d starting from the first character,

$leftIndex(t, d)$ – offset of the first occurrence of t in d starting from the last character,

$wikiCategories(t)$ – function that searches for the Wikipedia categories that are associated with candidate keyword t.

Frequency-based features
TF: Term frequency in the document.
TF-IDF: TF-IDF weight in the corpus, computed using the formulas $tf(t, d) = \frac{f(t,d)}{\max\{f(w,d):w \in d\}}$ and $idf(t, D) = \log \frac{|D|}{df(t,D)}$.
Ext. TF-IDF: Average TF-IDF weight computed using the top 5 documents returned by an indexed version of the external corpus. Note that this features only considers the title part of the external document and not its full content.
GDC: The Generalized Dice Coefficient [16] is a score that relies on lexical cohesion for estimating if an n-gram is a suitable keyword, it is calculated for a given document using $gdc(t, d) = \frac{|t| log_{10}(f(t,d))f(t,d)}{\sum_{w_i \in t} f(w_i,d)}$.

Word-based features
Length: Number of characters in the candidate keyword, $len(t)$.
Nb. Tokens: Number of tokens in the candidate keyword, $|t|$.
First word suffix: Boolean feature that is set to true when the three last characters of the first word in the candidate keyword belong to a closed list of values.
Last word suffix: Boolean feature that is set to true when the three last characters of the last word in the candidate keyword belong to a closed list of values.
Contains digit: Boolean feature that is set to true when the candidate contains a number.
First line overlap: Overlap score between the candidate keyword and the first line of the document,
$overlap(firstLine, t) = \frac{2*|firstLine \cap t|}{len(firstLine)+len(t)}$.

Position-based features
First occurrence: Normalized position of the first occurrence of the candidate in the document, $f_0(t, d) = \frac{rightIndex(t,d)}{len(d)}$.
Last occurrence: Normalized position of the last occurrence of the candidate in the document, $l_0(t, d) = \frac{leftIndex(t,d)}{len(d)}$.
Depth: Depth of the candidate keyword in the document,
$depth(t, d) = 1 - \frac{f_0(t,d)}{len(d)}$.
Span: Span of the candidate keyword in the document,
$span(t, d) = \frac{l_0(t,d)-f_0(t,d)}{len(d)}$.

Wikipedia-based features
Nb. Wikipedia resources: Number of Wikipedia resources found for the candidate keyword. In our case we only consider the top-5 most relevant resources, therefore this number is between [0-5].
Nb. Wikipedia categories: Total number of categories associated with a candidate keyword $|wikiCategories(t)|$
Number of categories in the keyword profile: The keyword profile contains information on the Wikipedia categories that have been selected by a user. This feature counts the number of categories in the user profile that are also in the categories linked with a candidate keyword ($wikiCategories(t)$). Details on the keyword profile are provided in Section 3.2.1.

To compute the values of Wikipedia derived features, we relied on the DBpedia Lookup Service [3]. DBpedia lookup Service is an index that allows to search for DBpedia URIs using a keyword[4]. In practice the function $wikiCategories(t)$ queries the DBpedia Lookup Services and considers all the categories for the top-5 results returned by the service.

3.2 Method for keyword ranking

The application of the classifier described in the previous section results in a set of candidate keywords that must be ranked according to their prominence in the document. Various methods have been proposed for this purpose. The most basic ones consist in: i) sorting the candidates according to the confidence score returned by the keyword classifier; ii) sorting the candidates according to their order of appearance in the document.

Other methods rely on ranking function [16], or semantic similarity measure such as [17]. Precisely, they compute a Google similarity distance between all distinct pairs of candidate keywords, then order the pairs in ascending order of their similarity scores. Finally the pairs having the lowest similarity score are ranked first. More recently [6] proposed to rank keywords by computing their TF-IDF score based on the titles of the articles in Wikipedia. In this article we propose to rank keywords using a learning-to-rank algorithm that uses features extracted from a user defined keyword profile.

We will now introduce the purpose of the keyword profile and then present the features used for training the ranking model.

3.2.1 Keyword profile

The purpose of this profile is to augment an initial set of terms provided by a user, for a given domain, with the knowledge contained in the Wikipedia categories. Our idea is to find other terms in that same domain by exploiting the information provided by the categories.

The profile construction procedure is as follows: i) the keyword profile is initially set to a list of terms, provided by the user, that somehow represents the domain knowledge the user wants to favor for the candidate keyword selection; ii) we retrieve the categories associated with each term in the list (as described in the function $wikiCategories$ defined in Section 3.1) and add them to the keyword profile; iii) we remove from the profile all categories which number of occurrences is lower than a threshold k ($k = 3$) and iv) for each remaining category we retrieve all the labels of the Wikipedia entries that are associated with this category and add them to the profile. Thus, the resulting keyword profile is the union of P_{cat}, the set of all categories found in Wikipedia, and P_{inst}, which is the set of instance labels found in the last step of the process.

For each candidate keyword, we define a profile overlap score (*$profileOverlap$*) based on its similarity with the categories in the profile (*$catSim$*) and its similarity with the labels of the Wikipedia entries that correspond to instances of these categories (*$instSim$*).

[3] `http://wiki.dbpedia.org/lookup/`
[4] A detailed example of the service output can be seen at `http://lookup.dbpedia.org/api/search.asmx/KeywordSearch?QueryString=un`.

These scores are computed as below:

$$catSim(t) = \frac{|P_{cat} \cap t_{cat}|}{\min\{len(P_{cat}), len(t_{cat})\}} \quad (1)$$

$$instSim(t) = \frac{|P_{inst} \cap t_{inst}|}{\min\{len(P_{inst}), len(t_{inst})\}} \quad (2)$$

$$profileOverlap(t) = \alpha * catSim + (1 - \alpha) * instSim \quad (3)$$

where t_{cat} is the set of categories found for a candidate keyword, that is $t_{cat} = wikiCategories(t)$, and t_{inst} is the set of all labels of the Wikipedia entries that are associated with the categories in t_{cat}. Finally α is a weighting factor arbitrarily set to 0.75 to give more importance to the category overlap score.

3.2.2 Keyword ranking model

Among previous studies using learning-to-rank methods, [18] is comparable to this work. We replaced the SVM by a Passive-Aggressive Perceptron trained with Combined Regression and Ranking method [19]. This method alternates between pairwise rank-based steps and standard stochastic gradient steps on single examples. The implementation is available at: `http://code.google.com/p/sofia-ml/`. Note that we did not used any parameter tuning for the ranking model.

To rank the candidate keywords, we added new classification features to those previously presented for the keyword identification step:

Keyword identification derived features:
TF; *TF-IDF*; *SPAN*; *DEPTH*; *First occurrence*; *Last occurrence*; *Nb. Tokens*; *Length*; *First line overlap*.
Additional ranking features:
Profile overlap score: Score obtained by comparing the keyword with the keyword profile (*c.f* Section 3.2.1).
Nb. categories in profile: Number of Wikipedia categories associated with the keyword that are also in the keyword-profile.
Ext. NDF: Normalized document frequency in the external corpus, $ndf(t, K) = \frac{df(t,K)}{|K|}$.
Index feedback: This feature aims at including a ranking information, provided by a search engine, for a given candidate keyword t and a document d. In details, we indexed the corpus with a search engine[5] and then we queried the search engine to obtain a weighted list of documents *docList* that contain this candidate. Finally we consider as value for the feature the rank of d in *docList*. In practice the size of *docList* is limited to 15 results.

For training the ranking model we used a ranking system based on four rankings as decribed below:

- 3: is associated with candidate keywords that have been selected as keyword by humans annotators;
- 2: is associated with candidate keywords that have not been selected by humans but belong to a Wikipedia category;
- 1: is associated with candidate keywords that have not been selected by humans but belong to an instance of a Wikipedia category;
- 0: for all other candidate keywords.

[5]We used Lucene.

The outcome of the ranking model on the test data consists of a list of weights computed by the model for all the entries in the test data. In our case this list is sorted in decreasing order to match with the ranking scale we have previously defined. In that way, the best candidate keywords should obtain the highest weights.

4. EVALUATION

Our evaluation had two objectives: verify the pertinence of the features that we propose for the keyword classification (Section 4.2) and measure how accurate is our extraction approach while using different corpora (Section 4.3).

4.1 Resources

Our experiments were based on the training data provided by the SemEval keyword dataset [15]. The dataset is composed of 284 scientific articles that have been annotated by both authors and readers. In total 2070 keywords were provided for training and 1443 keywords for testing. Note that only the stemmed keywords are provided for testing. In addition to this benchmark we conducted experiments on the Wiki20 corpus introduced in [20], which is composed of 20 computer science articles annotated by 15 teams of students.

As external corpus, we employed the TAC-KBP 2012 English corpus[6]. The corpus contains 3,8 million documents and is mainly composed of news articles ($\simeq 60\%$) and web documents ($\simeq 40\%$).

For constructing the keyword-profile, we used an initial set of keywords derived from Microsoft Academic Search engine: the search engine provides a list of top keywords in terms of number of publications and number of citations. Different filters can be applied to refine the domains and sub-domains of interest. In this study we automatically extracted the top 1000 keywords for all the 15 domains available. We removed duplicates and applied the procedure presented in Section 3.2.1 for constructing the keyword profile. In total, the profile is composed of 500 categories and 57281 labels of Wikipedia entries.

For POS-tagging of candidate keywords we employed the Stanford Core-NLP toolkit[7]. We relied on the implementation of learning algorithms in Weka[8] for testing various models for training our keyword classifier.

4.2 Keyword extraction

As documents used for experimentation are scientific articles, representative keywords are generally at the beginning of the documents which influences the evaluation process. To adapt our approach to this specificity, we discarded the acknowledgment and reference sections. Then we applied the candidate keyword generation procedure described in Section 3.1 to train the keyword classifier.

For constructing the training data we relied on the training data provided by the SemEval corpus: our set of positive examples is composed of all the keywords identified by the human annotators in the training data (in total 1288 keywords). For the set of negative examples, we considered all candidate keywords generated from the all the training documents and removed all the candidate keywords

[6]`http://www.nist.gov/tac/2012/KBP/data.html`
[7]`http://nlp.stanford.edu/software/corenlp.shtml`
[8]`http://www.cs.waikato.ac.nz/ml/weka/`

Table 1: Evaluation of the keyword classifier

Algorithm	R. (%)	P. (%)	F. (%)
Frequency-based features			
Meta bagging	57.9	75.2	65.4
Decision trees	56.7	75.4	65.3
Naive Bayes	42.5	85.0	56.7
+ Word-based features			
Meta bagging	71.4	76.8	74.0
Decision trees	68.5	78.9	73.3
Naive Bayes	50.4	82.9	62.7
+ Position-based features			
Meta bagging	76.3	80.1	78.2
Decision trees	75.4	78.2	76.8
Naive Bayes	58.9	82.2	68.7
+ Wikipedia-based features (all)			
Meta bagging	77.3	80.0	78.6
Decision trees	76.2	78.1	77.1
Naive Bayes	59.7	81.9	69.1

Table 2: Global evaluation of the keyword extraction using SemEval data

	Training data			Testing data		
Approach	R. (%)	P. (%)	F. (%)	R. (%)	P. (%)	F. (%)
RAKE baseline	11.9	11.6	11.8	10.6	10.7	10.7
Extractor_No_Profile	21.0	23.7	22.2	15.5	22.3	18.3
Extractor_Profile	21.4	24.0	22.7	18.0	20.8	19.3

Table 3: Global evaluation of the keyword extraction using Wiki20 data

Approach	R. (%)	P. (%)	F. (%)
RAKE baseline	10.2	2.9	4.5
Extractor_Profile	17.8	8.6	11.6

already in the positive set. As the number of negative examples is very high compared with the number of positive examples we randomly selected a subset of negative examples composed of 2576 candidate keywords (twice the number of positive examples). Concerning the tuning of the parameters, we used the default values provided by the Weka toolkit.

We report in Table 1 the results obtained with three algorithms and different feature sets. Note that the results were obtained using 5-fold cross validation and are reported in terms of recall (R.), precision (P.) and F-score (F.).

Table 1 shows that the best results are obtained using Meta bagging [21] algorithm across all the different feature sets. Also, the results show that we can achieve limited performance simply by using the frequency-based feature, around 65% using decision trees and Meta bagging. Among the different features, the word-based group has the highest impact: results are improved by 8% using Meta bagging and Decision trees. The results demonstrate that the Wikipedia-based features have a positive impact on the results (for all algorithms). Compared with considering only the frequency-based features, we observe that the F-score increases by an average of +12.4%. Following these results we selected the Meta bagging algorithm for the next processing steps.

4.3 Overall evaluation

In this section, we focus on the extrinsic evaluation of our approach. Precisely, we want to measure how good the keyword classifier and the ranking model perform when combined together. We are also interested in evaluating the impact of the keyword profile on the process. In this context we proposed as baseline a frequency-based approach founded on RAKE algorithm [12]: the RAKE baseline relies on a set of stop-words and word delimiters to split the document content and identify potential keywords. A score is then attributed to each candidate keyword depending on the frequency of its member words. In addition to the baseline, we provide the results of our approach i) without considering information from the keyword profile (*Extractor_No_Profile*) ii) and considering these information (*Extractor_Profile*).

Note that we applied the filters presented in Section 3.1 on the candidate keywords generated by the RAKE baseline. The ranking procedure used in the baseline simply returns the candidates according to their order of appearance in the document. We report in Table 2 the results for all three approaches using SemEval training and testing data and in Table 3, their results on the Wiki20 corpus. In both tables the results are reported in terms of recall (R.), precision (P.) and F-score (F.). Note that the results on SemEval are based on the top-15 keywords returned by each approach.

Table 2 shows that our keyword classifier outperforms the baseline on both training and testing sets. By comparing the baseline and our *Extractor_No_Profile* method we observe an improvement of +10.4% in terms of F-score on the training data and +7.6% on the testing data. The results also show that our approach is more sensitive than the baseline while moving from training data to testing data. On average the discrepancy is 3.6% in terms of F-score for both *Extractor_No_Profile* and *Extractor_Profile* methods. Concerning the keyword profile, the results demonstrate that it has a positive impact on the training data (+0.5% F-score) that is also confirmed on the testing data (+1.01% F-score). From a global point of view our results are lower than those of the top participant at the evaluation: [16] reported 27.5% F-score for the top 15 keywords. Compared with their approach, our method does not consider features on the structure of the document (in terms of abstract, introduction, etc.) that are quite important when dealing with scientific articles. Since our goal is to apply our method in various contexts, we experimented features that are generic and applicable on many different documents.

Table 3 shows the result of the RAKE baseline compared with the keyword classifier. We can observe that our approach outperforms the baseline (+7.1%). However, the F-score, compared to the results obtained on the SemEval dataset, is lower. This can be attributed to a loss of precision. It is important to notice that the keywords have been annotated with 15 teams, which made it difficult to reach high inter-annotator agreement: [20] underlined that the consistency among the different teams is low compared with professional indexers. Unfortunately [20] do not report results in terms of precision and recall, as their main interest is not keyword extraction.

5. CONCLUSIONS

In this paper, we presented an approach for automatic keyword extraction based on machine learning methods. We integrated information from multiple sources to enhance the keyword extraction process. We suggested to use Wikipedia-based features to improve the classification process of candidate keywords. We experimented the use of several features that can bring a small improvement in a keyword identification task.

In addition we proposed to use a keyword profile based on Wikipedia categories.A user could use such profile to integrate information on the domain knowledge that he wants to emphasize during both candidate keyword extraction and ranking process. Experiments have shown that the keyword profile is relevant, as a positive impact was observed during the global keyword extraction evaluation. Precisely, we tested our approach on the SemEval corpus and the Wiki20 corpus that are both composed of scientific articles. We compared our approach to a document-centered baseline and saw that it outperforms the baseline on both corpus.

6. ACKNOWLEDGMENTS

This work was supported by MITACS (Canada) and Messaging Architects.

7. REFERENCES

[1] Rao, W., Chen, L., Hui, P., Tarkoma, S.: Move: A large scale keyword-based content filtering and dissemination system. In: IEEE 32nd ICDS. (2012)

[2] Hammouda, K.M., Matute, D.N., Kamel, M.S.: Corephrase: keyphrase extraction for document clustering. In: Proceedings of MLDM'05, Springer-Verlag (2005)

[3] Charton, E., Camelin, N., Acuna-Agost, R., Gotab, P., Lavalley, R., Kessler, R., Fernandez, S.: Pré-traitements classiques ou par analyse distributionnelle: application aux méthodes de classification automatique déployées pour deft08. In: Actes DEFT08-TALNŠ08. (2008)

[4] Yih, W.t., Goodman, J., Carvalho, V.R.: Finding advertising keywords on web pages. In: Proceedings of WWW '06, USA, ACM (2006)

[5] Yang, S., Jin, J., Parag, J., Liu, S.: Contextual advertising for web article printing. In: Proceedings of DocEng '10, USA, ACM (2010)

[6] Vidal, M., Menezes, G.V., Berlt, K., de Moura, E.S., Okada, K., Ziviani, N., Fernandes, D., Cristo, M.: Selecting keywords to represent web pages using wikipedia information. In: Proceedings of WebMedia '12, USA, ACM (2012)

[7] Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: Kea: practical automatic keyphrase extraction. In: Proceedings of DL '99, USA, ACM (1999)

[8] Turney, P.D.: Learning algorithms for keyphrase extraction. Inf. Retr. **2** (2000)

[9] Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of EMNLP '03, ACL (2003)

[10] Zhang, C., Wang, H., Liu, Y., Wu, D., Liao, Y., Wang, B.: Automatic keyword extraction from documents using conditional random fields. Journal of Computational Information Systems (2008) 1169–1180

[11] Matsuo, Y., Ishizuka, M.: Keyword extraction from a single document using word co-occurrence statistical information. International Journal on Artificial Intelligence Tools **13** (2004) 157–169

[12] Stuart, R., Dave, E., Nick, C., Wendy, C.: Automatic Keyword Extraction from Individual Documents. In: Text Mining. John Wiley & Sons, Ltd (2010) 1–20

[13] Medelyan, O., Frank, E., Witten, I.H.: Human-competitive tagging using automatic keyphrase extraction. In: Proceedings of EMNLP '09, ACL (2009)

[14] Grineva, M., Grinev, M., Lizorkin, D.: Extracting key terms from noisy and multitheme documents. In: Proceedings of WWW '09, ACM (2009)

[15] Kim, S.N., Medelyan, O., Kan, M.Y., Baldwin, T.: Semeval-2010 task 5 : Automatic keyphrase extraction from scientific articles. In: Proceedings of the 5th International Workshop on Semantic Evaluation, Sweden, ACL (2010)

[16] Lopez, P., Romary, L.: Humb: Automatic key term extraction from scientific articles in grobid. In: Proceedings of the 5th International Workshop on Semantic Evaluation, Sweden, ACL (2010)

[17] Chen, P.I., Lin, S.J.: Automatic keyword prediction using google similarity distance. Expert Systems with Applications **37** (2010)

[18] Eichler, K., Neumann, G.: Dfki keywe: Ranking keyphrases extracted from scientific articles. In: Proceedings of the 5th International Workshop on Semantic Evaluation, ACL (2010)

[19] Sculley, D.: Combined regression and ranking. In: Proceedings of KDD '10, ACM (2010)

[20] Medelyan, O., Witten, I.H., Milne, D.: Topic indexing with Wikipedia. In: Proceedings of the Wikipedia and AI workshop at AAAI-08. (2008)

[21] Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140