Original scientific paper

Automatic Keyphrase Extraction from Croatian Newspaper Articles

Renee Ahel
Faculty of Electrical Engineering and Computing,
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
renee.ahel@gmail.com

Bojana Dalbelo Bašić
Faculty of Electrical Engineering and Computing,
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
bojana.dalbelo@fer.hr

Jan Šnajder
Faculty of Electrical Engineering and Computing,
University of Zagreb
Unska 3, 10000 Zagreb, Croatia
jan.snajder@fer.hr

Summary

Keyphrases provide a way to summarize documents and enable cross-category retrieval. The paper describes a robust system for automatic keyphrase extraction from newspaper articles in Croatian language. Keyphrase candidates are generated based on linguistic and statistical features, and naïve Bayes classifier is used to select the best keyphrases among the candidates. A prediction model is built using training documents with human-assigned keyphrases. System performance is measured on a corpus of newspaper articles, by comparing the automatically extracted keyphrases with those assigned by professional indexers. In absence of comparable results, we consider our results to be of modest performance.

Key words: keyphrase extraction, naïve Bayes classifier, Croatian language

Introduction

In recent decade, the number of digital documents is growing exponentially, caused by ever-growing use of computers in every aspect of human endeavor. Along with that growth, the need for efficient search, indexing, and categorization over those documents becomes ever more important. A practical way of summarizing document contents is to assign keyphrases. Keyphrases describe the content of the document, thus enabling the user to decide whether a particular document is relevant for his or her information need, without reading the entire document. Unfortunately, most authors assign keyphrases to their documents only when they are compelled to do so. Manual assignment of keyphrases is a tedious task, especially considering the ever-growing amount of documents. Thus, there is a great need for means of automatic keyphrase assignment.

Keyphrase assignment methods can be divided into two categories: *keyphrase assignment* and *keyphrase extraction*. Both methods revolve around the same fundamental problem of selecting the best keyphrases, and in both methods this problem is tackled as a machine learning problem – a chosen algorithm learns the “good keyphrase” concept using documents with manually pre-assigned keyphrases. Automatic keyphrase assignment is the closest to human keyphrase assignment in that keyphrases are chosen from a predefined thesaurus (a limited, possibly hierarchically organized set of possible keyphrases). Keyphrase extraction generates keyphrases from document text, rather than from a thesaurus. Since authors may assign keyphrases that do not exist in document text, the performance of extraction methods tends to be lower than that of assignment methods. Turney (2000) argues that 65%–90% of author assigned keyphrases appear verbatim in document text, implying a rather satisfactory upper bound on recall at 90%.

To the best of our knowledge, no previous keyphrase extraction or assignment approaches have been developed for Croatian language. This paper deals with keyphrase extraction method only, and applies it to the Croatian newspaper articles. For a more efficient keyphrase extraction, problems relating to inflectional morphological complexity of the Croatian language will also be addressed.

The rest of the paper is structured as follows. In Section 2 an overview of related work for keyword extraction for English language is given. Section 3 describes the four phases of our keyphrase extraction algorithm, while Section 4 shows the results of experimental evaluation. Section 5 concludes the paper and outlines future work.

Related work

The majority of work on keyword extraction deals with the English language. Turney (2000) used Quinlan’s C4.5 decision tree algorithm combined with bagging technique, as well as his own algorithm called GenEx. GenEx algorithm consists of a parameterized keyphrase extraction algorithm paired with Genitor genetic algorithm for parameter optimization. He applies GenEx on several corpora, ranging from e-mails and scientific papers to user and institutional web pages. Several variations of the C4.5 algorithm are applied by varying the number of trees and the proportion of sampled positives. Algorithm C4.5 with 50 bagged trees and 1% positive candidate sampling was shown to perform best, while GenEx algorithm has been shown to consistently outperform C4.5 on all tested corpora.

Witten et al. (1999) used a naïve Bayes classifier on a subset of corpora that was used by Turney as well as their own CSTR (Computer Science Technical Reports) corpus. Their so-called KEA algorithm outperforms Turney's on some corpora.

KEA++ is an improved version of the KEA algorithm devised by Medelyan and Witten (2006). KEA++ performs thesaurus-based term expansion using the Agrovoc thesaurus. On documents from agricultural domain, KEA++ significantly outperformed KEA.

Hulth (2003) used a rule induction system combined with bagging technique and POS tags, n-grams, and NP-chunks based methods on a corpus of scientific papers from the Inspec database. Results indicated that POS tags method outperforms other methods, while additional combination of all three methods helps reduce false positives.

Ercan and Cicekli (2007) used the C4.5 algorithm combined with bagging technique. They calculated candidate features using lexical chains and WordNet ontology. Their results indicated that the use of lexical chain-based features improves keyword extraction.

Work reported in this paper is inspired by KEA (Witten et al. 1999), mainly because this algorithm has shown good performance using a rather simple set of features. We improve on KEA's candidate generation and feature selection mechanisms, and additionally use POS tag filtering similar to that of Hulth (2003). Turney's approach is less adequate in our case because the GenEx algorithm was tailored for the English language, while C4.5 was in fact shown to perform worse than KEA. Other abovementioned approaches are yet unfeasible due to the lack of appropriate linguistic resources for Croatian language.

Keyphrase extraction

In this section, we describe the four phases of our keyphrase extraction algorithm: preprocessing, candidate generation, feature calculation, and learning and classification. The algorithm operates on documents that consist of a title, text body, pre-assigned keyphrases, and pre-assigned set of categories. Additional information source that we use is a predefined category taxonomy.

Pre-processing

Each individual document is pre-processed as follows. Document text is first divided into sections determined by the so-called *phrase boundaries* (punctuation marks, brackets, and non-printable characters such as new line, new paragraph, etc.). Phrase boundaries limit parts of document text within which keyphrase candidates are generated. Afterwards, text is tokenized based on regular expressions; tokens not consisting of at least one letter or digit are discarded and all tokens are case-folded. Tokens are lemmatized using the automatically acquired inflectional lexicon (Šnajder et al. 2008). Because in our case lemmatization does not disambiguate homographs, each token may have one or more lemmas assigned. Along with each lemma, a POS tag is assigned using the same lemmatizer. If the lemmatizer does not recognize a token, the original value of the token is assigned as the lemma, and a special POS tag "F" (lemmatization failed) is assigned. The set of possible POS tags is: N (noun), A (adjective), V (verb), X (stopword – pronoun, conjunction, preposition, interjection, number, particle), F (lemmatization failed). Stopwords are detected by comparing against a fixed list of stopwords for Croatian language. The same processing that is applied to document text body is performed on the pre-assigned keyphrases, as well as the title of the document. Additionally, for each token from the document text we check if it appears in (1) a name of a category from category taxonomy or (2) the document title.

Candidate generation

Next step is the keyphrase candidate generation, for which we use the previously determined phrase boundaries. Candidate is generated as a sequence of one, two, or three consecutive tokens within neighboring phrase boundaries. Since every token has one or more lemmas assigned, for each candidate all possible lemma combinations are generated, along with their POS tag combinations (POS patterns).

After all lemmatized forms for a candidate are generated, a POS pattern filter is applied. Only those candidates having at least one lemmatized form whose POS pattern passes the POS filter are retained; for these candidates, only lemma forms passing the POS filter are stored. POS patterns used in the POS filter are taken from (Petrović et al. 2009), where they were used to detect collocations; preliminary experiments have shown that these patterns will also be suitable for keyphrase extraction. POS filter patterns are given in Table 1.

Table 1: POS patterns used in POS filter

N, F
AN, NN, NF, FN
NXN, NAN, AAN, ANN, NNN, ANF, NXF, AFN, FNN, FXN, NFN, NNF

For each candidate that passed the POS filter, some further processing is performed. It is determined if the candidate appears in the category taxonomy; a candidate is considered to appear in the category taxonomy if at least one of its tokens appears in the category taxonomy. Similarly, a candidate is considered to appear in document title if at least one of its tokens appears in the title. Position of candidate appearance within document text is determined as the position of the first candidate token in the document.

Since candidates are generated within phrase boundaries, it is possible that on document level multiple instances of the same candidate are generated. Candidate instances are resolved using lemmatized forms of the candidates, otherwise

different inflectional forms of the same candidate would not count as distinct candidates.
In our case, since every candidate can have more than one lemmatized form, two candidates are considered to be instances of the same candidate if they share at least one common lemmatized form. All subsequent instances are replaced by the candidate instance that appears first in a document.
After candidate instances have been resolved, a match with pre-assigned keyphrases is performed. It is again based on comparison of corresponding lemmatized forms, using the abovementioned matching principle.
Note that only one lemmatized form match is sufficient to determine a match. Also note that pre-assigned keyphrases do not have to pass the POS filter, whereas the candidates do. Thus, some pre-assigned candidates will never match to a potential candidate if that candidate did not pass the POS filter. Nevertheless, insisting that candidates must pass the POS filter is a deliberate trade-off in which a small number of false negatives is traded for a more significant reduction in the number of false positives.

Feature calculation
After keyphrase candidates have been generated, each candidate is described using the following four features:

- relative first appearance in document text (*First*$_R$),
- *TFxIDF* value,
- a value that indicates whether the candidate appears in category taxonomy (*IsInCategory*),
- a value that indicates whether a candidate appears in document title (*IsInTitle*).

Relative first appearance in document text is determined using the following formula:

$$first_R(C) = \frac{first(C,D)}{size(D)},$$

where *first(P,D)* represents the position of first appearance of candidate *C* in text of the document *D*, and *size(D)* represents the total length of document *D*, measured in tokens.
TFxIDF is calculated as:

$$TFxIDF = \frac{freq(C,D)}{size(D)} \times \log_2 \frac{N}{df(C)},$$

where *freq(C,D)* represents the frequency of the candidate *C* in document *D*, value *df(C)* represents the number of documents in the entire corpus where candidate *C* appears, and *N* is the total number of documents.
The motivation behind the *IsInCategory* and *IsInTitle* features is that candiates are likely to be more descriptive – and thus better keyphrases – if they appear in category taxonomy or document titles, respectively.

Relative first appearance and *TFxIDF* are quantitative variables, which can pose a problem for the naïve Bayes classifier. Moreover, Witten et al. (1999) argue that discretizing quantitative variables improves prediction performance. We employ two means of discretization: percentile discretization (feature values are ordered ascending and divided into 100 bins containing equal numbers of candidates) and entropy-based discretization using the Minimum Description Length principle proposed by Fayyad and Irani (1993).

Learning and ranking

Application of the naïve Bayes classifier to the problem of keyphrase extraction using described features amounts to calculation of two posterior probabilities for each candidate *C*:

$$p(key \mid First_R(C), TFxIDF(C), IsInCategory(C), IsInTitle(C)) = \\ p(key)p(First_R(C) \mid key)p(TFxIDF(C) \mid key)p(IsInCategory(C) \mid key) \\ p(IsInTitle(C) \mid key)$$

(1)

$$p(\neg key \mid First_R(C), TFxIDF(C), IsInCategory(C), IsInTitle(C)) = \\ p(\neg key)p(First_R(C) \mid \neg key)p(TFxIDF(C) \mid \neg key)p(IsInCategory(C) \mid \neg key) \\ p(IsInTitle(C) \mid \neg key)$$

(2)

i.e., the probability that, given its feature values, candidate *C* is a keyphrase or is not a keyphrase, respectively. Value *p(key)* is the *a priori* probability that any given candidate is a keyphrase. Value $p(First_R(C)|key)$ is the conditional probability that a specific feature value $First_R(C)$ will appear, given that candidate *C* is a keyphrase; and similarly for other features. Formulae (1) and (2) derive from the Bayes theorem under the independence events assumption, which – despite being often theoretically unjustified – has been found to work well in practice.

The learning process consists of estimating, for each feature value, the probabilities necessary to calculate posterior probabilities given by (1) and (2). To this end, we use the so-called m-probability estimate (Mitchell 1997).

The ranking process for candidate *C* consists of calculating the posterior probabilities according to (1) and (2). Calculated probabilities are then normalized by their sum to unit scale. Candidates from the same document are ordered descending by the normalized value of (1) and descending by *TFxIDF*. Similar candidates are then removed from ordered list of candidates; two candidates are considered similar if they overlap in text, and in this case the lower-ranked of the two candidates is removed from the list. Resulting ranked list is the final list of extracted keyphrases. From this list we take the *N* top-ranked candidates as the document keyphrases.

Corpora

Two corpora of Croatian newspaper articles have been made available by the Croatian News Agency (HINA): *October 4457* and *January 4532*. Documents in both corpora have pre-assigned keyphrases; from these phrases, we filtered out those that do not appear in text. Basic statistics for the two corpora is summarized in Table 2.

Table 2: Basic statistics for corpora *October 4457* and *January 4532*

	October 4457	**January 4532**
Number of documents	3905	4521
Average document length	525	335
Average number of candidates per document	235	153
Average number of key-phrases per document	2	2

Deficiencies of the available corpora

Human indexers have assigned keyphrases to documents regardless of whether the phrases appear in document text. In our case – because we are addressing the task of keyword extraction rather than keyword assignment – we have filtered out the keyphrases that do not appear in document text. This reduced the total number of keyphrases by 57%. Discarding keyphrases that do not appear in text inevitably deteriorates the quality of the training corpora. In some cases, more descriptive keyphrases will be removed, while the less descriptive keyphrases remain, merely because they appear in document text. Consequently, some meaningful automatically extracted keyphrases will not be matched against less descriptive pre-assigned keyphrases.

Another issue worth mentioning is that, in order to objectively judge system's performance, an evaluation of inter-annotator agreement should be carried out. We leave this important issue for future work.

Inconsistency is yet another curious characteristic of the described corpora, reflected by the fact that 63% of all keyphrases have been assigned to a single document. Not only does this deteriorate the quality of the training corpora, but it also raises doubts about the usefulness of the pre-assigned keyphrases as a means for cross-category search.

Experimental training set

From the described corpora we have compiled an experimental dataset for training and evaluation. The set consists of 200 newspaper articles with pre-assigned keyphrases that appear in document text. On average, there are 6.5 such keyphrases per document and 370 keyphrase candidates per document. Because current version of our system generates candidates only up to length three, pre-assigned keyphrases longer than that were discarded from the training set; this amounts to 1.7% of the pre-assigned keyphrases.

Results and discussion

In the three experiments that follow, performance is measured in terms of the F_1 measure with 10-fold cross validation on the training set. Because cross validation is used, results are expressed as a mean value of all ten iterations.
First experiment compares the performance of two basic configurations, each using a different discretization method. Results for some selected numbers of extracted keyphrases are given in Table 3. Performance rises steeply until seven extracted keyphrases, and more slowly afterwards. Best performance is achieved for 10 keyphrases with MDL discretization and for 12 keyphrases with percentile discretization. Results show that MDL discretization consistently outperforms percentile discretization.

Table 3: Performance of percentile discretization compared to MDL discretization

	Extracted keyphrases	**Precision (%)**	**Recall (%)**	**F_1 (%)**
MDL	1	**22.0**	3.4	5.9
	7	13.4	14.5	13.9
	10	12.5	19.3	**15.1**
	15	10.4	**24.1**	14.5
Percentile	1	**18.5**	2.9	5.0
	7	12.1	13.1	12.6
	12	10.5	19.5	**13.7**
	15	9.4	**21.8**	13.1

The second experiment examines the effect of additional POS filtering of the candidates. Prior to this, we have carried out an analysis of POS patterns of all candidates from corpora *October 4457* and *January 4532* (cf. Section 4). Results of this analysis, given in Table 4, reveal that by filtering out the candidates that do not match the POS patterns N, AN, NN, NXN we can discard 30% negative candidates, while discarding only 7.5% positive candidates.

Table 4: POS patterns of keyphrase/not keyphrase candidates

POS pattern	Keyphrase (%)	Not a keyphrase (%)	Total (%)
N	49.77	38.75	38.87
AN	28.17	13.26	13.42
NN	11.58	10.41	10.42
F	1.91	9.72	9.64
NXN	2.98	7.57	7.52
NAN	1.56	3.22	3.2
Other	4.02	17.06	16.92

Performance of two basic configurations combined with the described POS filtering is shown in Table 5. Results reveal that additional POS filtering consistently improves performance. E.g., for 10 extracted keyphrases with MDL discretization the improvement is 2.1%, while for 12 extracted keyphrases with percentile discretization the improvement is 1.7%.

Table 5: Performance with additional POS filtering of the candidates

	Extracted keyphrases	Precision (%)	Recall (%)	F_1 (%)	% change F_1
MDL + POS filtering	1	**23.5**	4.0	6.9	14
	7	14.9	17.9	16.3	14
	10	13.7	**23.4**	**17.2**	12
Percentile + POS filtering	1	**18.0**	3.1	5.3	6
	7	13.2	15.9	14.4	13
	12	11.4	**23.5**	**15.4**	11

The best overall result from the above experiments is obtained for 10 keyphrases using MDL discretization with POS filtering. Examples of keyphrases extracted from two documents using that configuration are given in Table 6 (true positives are underlined).

Third experiment is an ablation study: we measure the influence of each feature on performance by holding out one feature and doing keyphrase extraction using the remaining features. Experiment is carried out using the previously established best configuration (MDL + POS, 10 keyphrases). Results are given in Table 7.

Except for *IsInTitle* feature, which seems not to contribute to the performance, all other features seem to positively affect the performance. The improvement is, however, statistically significant at the 0.05 level on for the *TFxIDF* feature.

Table 6: Extracted keyphrases compared to pre-assigned keyphrases

Document title	Pre-assigned keyphrases	Extracted keyphrases
Lijevo-desna nerazumi-	*ljevica*	*poimanju*
jevanja	*politički život*	*suvereniteta*
	vrijednosti	*stranke*
	socijalna država	*politika*
	socijaldemokrati	*različitim poimanjem*
	liberali	*predizbornu kampanju*
	socijalna politika	*zakonitosti*
	crkveni socijalni nauk	*liberale*
	ekonomska politika	*socijaldemokrate*
	suverenitet	*konzervativce*
	zakonitost	
EU lansirala petogodišnji	*akcijski plan*	*petogodišnji plan*
plan sigurnosti	*sigurnost*	*sigurnosti*
	imigracijska politika	*akcijski plan*
	pravosuđe	*imigracijske politike*
	granična kontrola	*pravosuđa i sigurnosti*
	međunarodna suradnja	*europsku sigurnost*
		suradnje između zemalja
		terorizma
		sigurnost granica
		slobode i sigurnosti

Table 7: Influence of each feature on performance of the best configuration (MDL + POS, 10 keyphrases)

	Precision (%)	Recall (%)	F_1 (%)	% change F_1
IsInCategory	13.4	22.9	16.9	-2
IsInTitle	**13.7**	**23.5**	**17.3**	1
First_R	12.5	21.4	15.7	-9
TFxIDF	10.5	18.0	13.2	-23

Conclusion

Keyphrase extraction is a practical way to summarize document contents. Doing it manually on a large number of documents is a tedious task. In this paper, an algorithm for keyphrase extraction in Croatian language has been described and evaluated. In absence of comparable results, we consider our results to be of modest performance. This is mainly due to performance measure being based on pre-assigned keyphrases – though these were assigned by professional indexers, we suspect that the inter-annotator agreement might have been low. Furthermore, we have determined that documents from training corpora had inconsistently assigned keyphrases, which certainly negatively affects the performance. Nevertheless, results suggest that despite these deficiencies, it is possible to improve the extraction performance.

Future work can be focused on determining a more suitable performance measure, improving the quality of training corpora, as well as applying methods for dealing with the class disproportion problem.

Acknowledgments

This work has been supported by the Ministry of Science, Education and Sports, Republic of Croatia and under the Grant 036-1300646-1986. The authors are grateful to the Croatian News Agency (HINA) for making available the newspaper corpus.

References

Ercan, Gonenc; Cicekli, Ilyas. Using Lexical Chains for Keyword Extraction. // *Information Processing and Management.* 43 (2007), 6; 1705–1714

Fayyad, Usama M.; Irani, Keki B. Multi-interval Discretization of Continuous-valued Attributes for Classification Learning. *// Proceedings of the 11th International Joint Conference on Artificial Intellingence (IJCAI'93) /* Bajcsy, R. (ed.). Chambery, France : Morgan Kaufmann, 1993, 1022–1027

Frank, Eibe; Paynter, Gordon W.; Witten, Ian H.; Gutwin, Carl. Domain-specific Keyphrase Extraction. *// Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI '99) /* Dean, T. (ed.). Stockholm, Sweden : Morgan Kaufmann, 1999, 668–673

Hulth, Annette. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. *// Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'03) /* Sapporo, Japan : ACL Publications, 2003, 216–223

Hulth, Annette. Reducing False Positives by Expert Combination in Automatic Keyword Indexing. *// Proceedings of the Conference on Recent Advances in Natural Language Processing (RANLP'03) /* Borovets, Bulgaria : 2003, 216 – 223

Medelyan, Olena; Witten, Ian H. Thesaurus Based Automatic Keyphrase Indexing. *// Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries /* Chapel Hill, NC, USA : ACM Press, 2006, 296–297

Medelyan, Olena; Witten, Ian H. Thesaurus-based Index Term Extraction for Agricultural Documents. 09.05.2005. http://www.cs.waikato.ac.nz/~ihw/papers/05-OM-IHW-Agri-Docs.pdf (05.07.2009.)

Mitchell, Tom M. Machine Learning. New York : McGraw – Hill, 1997

Petrović, Saša; Šnajder, Jan; Dalbelo Bašić, Bojana. Extending Lexical Association Measures for Collocation Extraction. *// Computer Speech and Language.* (2009); doi:10.1016/j.csl.2009.06.001

Šnajder, Jan; Dalbelo Bašić, Bojana; Tadić, Marko. Automatic Acquisition of Inflectional Lexica for Morphological Normalisation. *// Information Processing and Management*. 44 (2008), 5; 1720–1731

Turney, Peter D. Learning to Extract Keyphrases from Text. // *Information Retrieval*. 4 (2000), 2; 303–336

Witten, Ian H.; Paynter, Gordon W.; Frank, Eibe; Gutwin, Carl. Kea: Practical Automatic Keyword Extraction. // *Proceedings of Digital Libraries 99 (DL '99)* / Rowe, N.; Fox, E. A. (ed.). Berkeley, CA, USA : ACM Press, 1999, 254–255