GPKEX: Genetically Programmed Keyphrase Extraction from Croatian Texts

Marko Bekavac and **Jan Šnajder**
University of Zagreb, Faculty of Electrical Engineering and Computing
Text Analysis and Knowledge Engineering Lab
Unska 3, 10000 Zagreb, Croatia
{marko.bekavac2,jan.snajder}@fer.hr

Abstract

We describe GPKEX, a keyphrase extraction method based on genetic programming. We represent keyphrase scoring measures as syntax trees and evolve them to produce rankings for keyphrase candidates extracted from text. We apply and evaluate GPKEX on Croatian newspaper articles. We show that GPKEX can evolve simple and interpretable keyphrase scoring measures that perform comparably to more complex machine learning methods previously developed for Croatian.

1 Introduction

Keyphrases are an effective way of summarizing document contents, useful for text categorization, document management, and search. Unlike *keyphrase assignment*, in which documents are assigned keyphrases from a predefined taxonomy, *keyphrase extraction* selects phrases from the text of the document. Extraction is preferred in cases when a taxonomy is not available or when its construction is not feasible, e.g., if the set of possible keyphrases is too large or changes often. Manual keyphrase extraction is extremely tedious and inconsistent, thus methods for automatic keyphrase extraction have attracted a lot of research interest.

In this paper we describe GPKEX, a keyphrase extraction method based on genetic programming (GP), an evolutionary optimization technique inspired by biological evolution (Koza and Poli, 1992). GP is similar to genetic algorithms except that the individual solutions are expressions, rather than values. We use GP to evolve keyphrase scoring measures, represented as abstract syntax trees. The advantage of using GP over black-box machine learning methods is in the interpretability of the results: GP yields interpretable expressions, revealing the relevant features and their relationships, thus offering some insight into keyphrase usage. Furthermore, GP can evolve simple scoring measures, providing an efficient alternative to more complex machine learning methods.

We apply GPKEX to Croatian language and evaluate it on a dataset of newspaper articles with manually extracted keyphrases. Our results show that GPKEX performs comparable to previous supervised and unsupervised approaches for Croatian, but has the advantage of generating simple and interpretable keyphrase scoring measures.

2 Related Work

Keyphrase extraction typically consist of two steps: candidate extraction and candidate scoring. Supervised approaches include decision tree models (Turney, 1999; Ercan and Cicekli, 2007), naïve Bayes classifier (Witten et al., 1999; McCallum and Nigam, 1998; Frank et al., 1999), and SVM (Zhang et al., 2006). Unsupervised approaches include clustering (Liu et al., 2009), graph-based methods (Mihalcea and Tarau, 2004), and language modeling (Tomokiyo and Hurst, 2003). Many more methods were proposed and evaluated within the SemEval shared task (Kim et al., 2010). Recent approaches (Jiang et al., 2009; Wang and Li, 2011; Eichler and Neumann, 2010) acknowledge keyphrase extraction as a highly subjective task and frame it as a learning-to-rank problem.

Keyphrase extraction for Croatian has been addressed in both supervised and unsupervised setting. Ahel et al. (2009) use a naïve Bayes classifier with phrase position and tf-idf (term frequency/inverse document frequency) as features. Saratlija et al. (2011) use distributional semantics to build topically related word clusters, from which they extract keywords and expand them to keyphrases. Mijić et al. (2010) use filtering based on morphosyntactic tags followed by tf-idf scoring.

Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing, pages 43–47,
Sofia, Bulgaria, 8-9 August 2013.

To the best of our knowledge, GPKEX is the first application of GP to keyphrase extraction. Although we essentially approach the problem as a classification task (we train on binary relevance judgments), GPKEX produces continuous-valued scoring measures, thus keyphrases can eventually be ranked and evaluated in a rank-based manner.

3 GPKEX

GPKEX (Genetically Programmed Keyphrase Extraction) consists of two steps: keyphrase candidate extraction and the genetic programming of keyphrase scoring measures (KSMs).[1]

3.1 Step 1: Keyphrase candidate extraction

Keyphrase candidate extraction starts with text preprocessing followed by keyphrase feature extraction. A keyphrase candidate is any sequence of words from the text that (1) does not span over (sub)sentence boundaries and (2) matches any of the predefined POS patterns (sequences of POS tags). The POS patterns are chosen based on the analysis of the training set (cf. Section 4).

After the candidates have been extracted, each candidate is assigned 11 features. We distinguish between three groups of features. The first group are the frequency-based features: the relative term frequency (the ratio between the number of phrase occurrences in a document and the total number of phrases in the document), inverse document frequency (the ratio between the total number of documents in the training set and the number of documents in which the phrase occurs), and the tf-idf value. These features serve to eliminate the irrelevant and non-discriminative phrases. The second group are the position-based features: the position of the first occurrence of a phrase in the text (i.e., the number of phrases in the text preceding the first occurrence of the candidate phrase), the position of the last occurrence, the occurrence in document title, and the number of occurrences in the first, second, and the last third of the document. These features serve to capture the relation between phrase relevance and the distribution of the phase within the document. The last group of features concerns the keyphrase surface form: its length and the number of discriminative words it contains (these being defined as the 10 words from the document with the highest tf-idf score).

[1] GPKEX is freely available for download from `http://takelab.fer.hr/gpkex`

3.2 Step 2: Genetic programming

Genetic expressions. Each keyphrase scoring measure (KSM) corresponds to one genetic expression, represented as a syntax tree (see Fig. 1). We use the above-described keyphrase features as outer nodes of an expression. For inner nodes we use binary ($+$, $-$, $\times$, and $/$) and unary operators ($\log \cdot$, $\cdot \times 10$, $\cdot / 10$, $1/\cdot$). We randomly generate the initial population of KSMs and use fitness-proportionate selection to guide the evolution process.

Fitness function. The fitness function scores KSMs according to their ability to extract correct keyphrases. We measure this by comparing the extracted keyphrases against the gold-standard keyphrases (cf. Section 4). We experimented with a number of fitness functions; simple functions, such as Precision at n (P@n) or Mean Reciprocal Rank (MRR), did not give satisfactory results. Instead, we define the fitness of a KSM s as

$$f(s) = \frac{1}{|D|} \sum_{d \in D} \begin{cases} \frac{|C_d^k|}{minRank(C_d^k)} & C_d^k \neq \emptyset, \\ \frac{1}{minRank(C_d^\infty)} & \text{otherwise} \end{cases} \quad (1)$$

where D is the set of training documents, C_d^k is the set of correct keyphrases within top k-ranked keyphrases extracted from document $d \in D$, and $minRank(C_d^k)$ is the highest rank (the smallest number) of keyphrase from set C_d^k. Parameter k defines a cutoff threshold, i.e., keyphrase ranked below rank k are discarded. If two KSMs extract the same number of correct keyphrases in top k results, the one with the highest-ranked correct keyphrase will be scored higher. To ensure that the gradient of the fitness function is non-zero, a KSM that extracts no correct keyphrases within the first k results is assigned a score based on the complete set of correctly extracted keyphrases (denoted C_d^∞). The fitness scores are averaged over the whole document collection. Based on preliminary experiments, we set the cutoff value to $k = 15$.

Parsimony pressure. Supervised models often face the problem of overfitting. In GP, overfitting is typically controlled by parsimony pressure, a regularization term that penalizes complex expressions. We define the regularized fitness function as

$$f_{reg} = \frac{f}{1 + N/\alpha} \quad (2)$$

where f is the non-regularized fitness function given by (1), N is the number of nodes in the expression, and parameter α defines the strength of

parsimony pressure. Note that in both regularized and non-regularized case we limit the size of an expression to a maximum depth of 17, which is often used as the limit (Riolo and Soule, 2009).

Crossover and mutation. Two expressions chosen for crossover exchange subtrees rooted at random nodes, resulting in a child expression with parts from both parent expressions. We use a population of 500 expressions and limit the number of generations to 50, as we observe that results stagnate after that point. To retain the quality of solution throughout the generations, we employ the elitist strategy and copy the best-fitted individual into the next generation. Moreover, we use mutation to prevent early local optimum trapping. We implement mutation as a randomly grown subtree rooted at a randomly chosen node. Each expression has a 5% probability of being mutated, with 10% probability of mutation at inner nodes.

4 Evaluation

Data set and preprocessing. We use the dataset developed by Mijić et al. (2010), comprising 1020 Croatian newspaper articles provided by the Croatian News Agency. The articles have been manually annotated by expert annotators, i.e., each document has an associated list of keyphrases. The number of extracted keyphrases per document varies between 1 and 7 (3.4 on average). The dataset is divided in two parts: 960 documents each annotated by a single annotator and 60 documents independently annotated by eight annotators. We use the first part for training and the second part for testing.

Based on dataset analysis, we chose the following POS patterns for keyphrase candidate filtering: N, AN, NN, NSN, V, U (N – noun, A – adjective, S – preposition, V – verb, U – unknown). Although a total of over 200 patterns would be needed to cover all keyphrases from the training set, we use only the six most frequent ones in order to reduce the number of candidates. These patterns account for cca. 70% of keyphrases, while reducing the number of candidates by cca. 80%. Note that we chose to only extract keyphrases of three words or less, thereby covering 93% of keyphrases. For lemmatization and (ambiguous) POS tagging, we use the inflectional lexicon from Šnajder et al. (2008), with additional suffix removal after lemmatization.

Evaluation methodology. Keyphrase extraction is a highly subjective task and there is no agreed-upon evaluation methodology. Annotators are often inconsistent: they extract different keyphrases and also keyphrases of varying length. What is more, an omission of a keyphrase by one of the annotators does not necessarily mean that the keyphrase is incorrect; it may merely indicate that it is less relevant. To account for this, we use rank-based evaluation measures. As our method produces a ranked list of keyphrases for each document, we can compare this list against a gold-standard keyphrase ranking for each document. We obtain the latter by aggregating the judgments of all annotators; the more annotators have extracted a keyphrase, the higher its ranking will be.[2] Following Zesch and Gurevych (2009), we consider the morphological variants when matching the keyphrases; however, we do not consider partial matches.

To evaluate a ranked list of extracted keyphrases, we use the generalized average precision (GAP) measure proposed by Kishida (2005). GAP generalizes average precision to multi-grade relevance judgments: it takes into account both precision (all correct items are ranked before all incorrect ones) and the quality of ranking (more relevant items are ranked before less relevant ones).

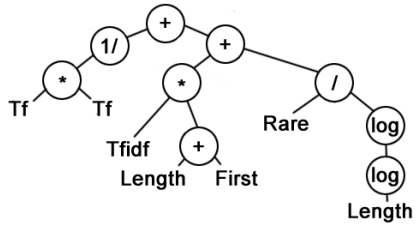
Another way of evaluating against keyphrases extracted by multiple annotators is to consider the different levels of agreement. We consider as *strong agreement* the cases in which a keyphrase is extracted by at least five annotators, and as *weak agreement* the cases in which at least two annotators have extracted a keyphrase. For both agreement levels separately, we compare the extracted keyphrases against the manually extracted keyphrases using rank-based IR measures of Precision at Rank 10 (P@10) and Recall at Rank 10 (R@10). Because GP is a stochastic algorithm, to account for randomness we made 30 runs of each experiment and report the average scores. On these samples, we use the unpaired t-test to determine the significance in performance differences. As baseline to compare against GPKEX, we use keyphrase extraction based on tf-idf scores (with the same preprocessing and filtering setup as for GPKEX).

Tested configurations. We tested four evolution configurations. Configuration A uses the parameter setting described in Section 3.2, but without parsimony pressure. Configurations B and C use parsimony pressure defined by (2), with $\alpha = 1000$

[2]The annotated dataset is available under CC BY-NC-SA license from `http://takelab.fer.hr/gpkex`

Config.	GAP	Strong agreement P@10	Strong agreement R@10	Weak agreement P@10	Weak agreement R@10
A	**13.0**	**8.3**	28.7	**28.7**	8.4
B	12.8	8.2	**30.2**	28.4	**8.5**
C	12.5	7.7	27.3	27.3	7.7
D	9.9	5.1	25.9	20.4	7.3
tf-idf	7.4	5.8	22.3	21.5	12.4
UKE	6.0	5.8	32.6	15.3	15.8

Table 1: Keyphrase ranking results.



$$\frac{1}{\mathsf{Tf}*\mathsf{Tf}} + \mathsf{Tfidf} * (\mathsf{Length} + \mathsf{First}) + \frac{\mathsf{Rare}}{\log(\log \mathsf{Length})}$$

Figure 1: The best-performing KSM expression.

and $\alpha = 100$, respectively. Configuration D is similar to A, but uses all POS patterns attested for keyphrases in the dataset.

Results. Results are shown in Table 1. Configurations A and B perform similarly across all evaluation measures (pairwise differences are not significant at $p<0.05$, except for R@10) and outperform the baseline (differences are significant at $p<0.01$). Configuration C is outperformed by configuration A (differences are significant at $p<0.05$). Configuration D outperforms the baseline, but is outperformed by other configurations (pairwise differences in GAP are significant at $p<0.05$), indicating that conservative POS filtering is beneficial. Since A and B perform similar, we conclude that applying parsimony pressure in our case only marginally improved GAP (although it has reduced KSM size from an average 30 nodes for configuration A to an average of 20 and 9 nodes for configurations B and C, respectively). We believe there are two reasons for this: first, the increase in KSM complexity also increases the probability that the KSM will be discarded as not computable (e.g., the right subtree of a '/' node evaluates to zero). Secondly, our fitness function is perhaps not fine-grained enough to allow more complex KSMs to emerge gradually, as small changes in keyphrase scores do not immediately affect the value of the fitness function.

In absolute terms, GAP values are rather low. This is mostly due to wrong ranking, rather than the omission of correct phrases. Furthermore, the precision for strong agreement is considerably lower than for weak agreement. This indicates that GPKEX often assigns high scores to less relevant keyphrases. Both deficiencies may be attributed to the fact that we do not learn to rank, but train on dataset with binary relevance judgments.

The best-performing KSM from configuration A is shown in Fig. 1 (simplified form). Length is the length of the phrase, First is the position of the first occurrence, and Rare is the number of discriminative words in a phrase (cf. Section 3.1). Tfidf, First, and Rare features seem to be positively correlated with keyphraseness. This particular KSM extracts on average three correct keyphrases (weak agreement) within the first 10 results.

Our results are not directly comparable to previous work for Croatian (Ahel et al., 2009; Mijić et al., 2010; Saratlija et al., 2011) because we use a different dataset and/or evaluation methodology. However, to allow for an indirect comparison, we re-evaluated the results of unsupervised keyphrase extraction (UKE) from Saratlija et al. (2011); we show the result in the last row of Table 1. GPKEX (configuration A) outperforms UKE in terms of precision (GAP and P@10), but performs worse in terms of recall. In terms of F1@10 (harmonic mean of P@10 and R@10), GPKEX performs better than UKE at the strong agreement level (12.9 vs. 9.9), but worse at the weak agreement level (13.0 vs. 15.6). For comparison, Saratlija et al. (2011) report UKE to be comparable to supervised method from Ahel et al. (2009), but better than the tf-idf extraction method from Mijić et al. (2010).

5 Conclusion

GPKEX uses genetically programmed scoring measures to assign rankings to keyphrase candidates. We evaluated GPKEX on Croatian texts and showed that it yields keyphrase scoring measures that perform comparable to other machine learning methods developed for Croatian. Thus, scoring measures evolved by GPKEX provide an efficient alternative to these more complex models. The focus of this work was on Croatian, but our method could easily be applied to other languages as well.

We have described a preliminary study. The next step is to apply GPKEX to directly learn keyphrase ranking. Using additional (e.g., syntactic) features might further improve the results.

Acknowledgments

This work has been supported by the Ministry of Science, Education and Sports, Republic of Croatia under the Grant 036-1300646-1986. We thank the reviewers for their constructive comments.

References

Renee Ahel, B Dalbelo Bašic, and Jan Šnajder. 2009. Automatic keyphrase extraction from Croatian newspaper articles. *The Future of Information Sciences, Digital Resources and Knowledge Sharing*, pages 207–218.

Kathrin Eichler and Günter Neumann. 2010. DFKI KeyWE: Ranking keyphrases extracted from scientific articles. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 150–153. Association for Computational Linguistics.

Gonenc Ercan and Ilyas Cicekli. 2007. Using lexical chains for keyword extraction. *Information Processing & Management*, 43(6):1705–1714.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of IJCAI '99*, pages 668–673. Morgan Kaufmann Publishers Inc.

Xin Jiang, Yunhua Hu, and Hang Li. 2009. A ranking approach to keyphrase extraction. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 756–757. ACM.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. SemEval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 21–26. Association for Computational Linguistics.

Kazuaki Kishida. 2005. *Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments*. National Institute of Informatics.

John R. Koza and Riccardo Poli. 1992. *Genetic Programming: On the programming of computers by Means of Natural Selection*. MIT Press.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP 2009*, pages 257–266, Singapore. ACL.

Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for Naïve Bayes text classification. In *AAAI-98 workshop on learning for text categorization*, pages 41–48. AAAI Press.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into texts. In *Proceedings of EMNLP*, volume 4. Barcelona, Spain.

Jure Mijić, B Dalbelo Bašic, and Jan Šnajder. 2010. Robust keyphrase extraction for a large-scale Croatian news production system. In *Proceedings of FASSBL*, pages 59–66.

Rick Riolo and Terence Soule. 2009. *Genetic Programming Theory and Practice VI*. Springer.

Josip Saratlija, Jan Šnajder, and Bojana Dalbelo Bašić. 2011. Unsupervised topic-oriented keyphrase extraction and its application to Croatian. In *Text, Speech and Dialogue*, pages 340–347. Springer.

Jan Šnajder, Bojana Dalbelo Bašić, and Marko Tadić. 2008. Automatic acquisition of inflectional lexica for morphological normalisation. *Information Processing & Management*, 44(5):1720–1731.

Takashi Tomokiyo and Matthew Hurst. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 33–40. Association for Computational Linguistics.

Peter Turney. 1999. Learning to extract keyphrases from text. Technical report, National Research Council, Institute for In- formation Technology.

C. Wang and S. Li. 2011. CoRankBayes: Bayesian learning to rank under the co-training framework and its application in keyphrase extraction. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2241–2244. ACM.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM.

Torsten Zesch and Iryna Gurevych. 2009. Approximate matching for evaluating keyphrase extraction. In *Proceedings of the 7th International Conference on Recent Advances in Natural Language Processing*, pages 484–489.

Kuo Zhang, Hui Xu, Jie Tang, and Juanzi Li. 2006. Keyword extraction using support vector machine. In *Advances in Web-Age Information Management*, volume 4016 of *LNCS*, pages 85–96. Springer Berlin / Heidelberg.