DKPro Keyphrases: Flexible and Reusable Keyphrase Extraction Experiments

Nicolai Erbs[†], Pedro Bispo Santos[†], Iryna Gurevych[†‡] and Torsten Zesch[§‡]

† UKP Lab, Technische Universität Darmstadt
‡ Information Center for Education, DIPF, Frankfurt
§ Language Technology Lab, University of Duisburg-Essen
http://www.ukp.tu-darmstadt.de

Abstract

DKPro Keyphrases is a keyphrase extraction framework based on UIMA. It offers a wide range of state-of-the-art keyphrase experiments approaches. At the same time, it is a workbench for developing new extraction approaches and evaluating their impact. DKPro Keyphrases is publicly available under an open-source license.[1]

1 Introduction

Keyphrases are single words or phrases that provide a summary of a text (Tucker and Whittaker, 2009) and thus might improve searching (Song et al., 2006) in a large collection of texts. As manual extraction of keyphrases is a tedious task, a wide variety of keyphrase extraction approaches has been proposed. Only few of them are freely available which makes it hard for researchers to replicate previous results or use keyphrase extraction in some other application, such as information retrieval (Manning et al., 2008), or question answering (Kwok et al., 2001).

In this paper, we describe our keyphrase extraction framework called DKPro Keyphrases. It integrates a wide range of state-of-the-art approaches for keyphrase extraction that can be directly used with limited knowledge of programming. However, for developers of new keyphrase extraction approaches, DKPro Keyphrases also offers a programming framework for developing new extraction algorithms and for evaluation of resulting effects. DKPro Keyphrases is based on the Unstructured Information Management Architecture (Ferrucci and Lally, 2004), which provides a rich source of libraries with preprocessing components.

[1] http://code.google.com/p/dkpro-keyphrases/

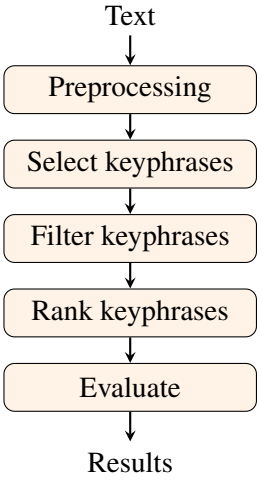

Figure 1: Architecture overview of DKPro Keyphrases

2 Architecture

The architecture of DKPro Keyphrases models the five fundamental steps of keyphrase extraction: (i) Reading of input data and enriching it with standard linguistic preprocessing, (ii) selecting phrases as keyphrase candidates based on the preprocessed text, (iii) filtering selected keyphrases, (iv) ranking remaining keyphrases, and (v) evaluating ranked keyphrases against a gold standard. This process is visualized in Figure 1. In this section, we will describe details of each step, including components already included in DKPro Keyphrases.

2.1 Preprocessing

DKPro Keyphrases relies on UIMA-based preprocessing components developed in the natural language processing framework DKPro Core (Gurevych et al., 2007; Eckart de Castilho and Gurevych, 2009). Thus, a wide range of linguistic preprocessing components are readily available such as word segmentation, lemmatization, part-of-speech tagging, named entity recognition, syn-

Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 31–36,
Baltimore, Maryland USA, June 23-24, 2014.

tactic parsing, or co-reference resolution.

2.2 Selecting Keyphrases

In this step, DKPro Keyphrases selects all phrases as keyphrases that match user-specified criteria. A criterium is typically a linguistic type, e.g. tokens, or more sophisticated types such as noun phrases. The resulting list of keyphrases should cover all gold keyphrases and at the same time be as selective as possible. We use the following sentence with the two gold keyphrases "dog" and "old cat" as a step through example:

> A [dog] chases an [old cat] in my garden.

Taking all uni- and bi-grams as keyphrases will easily match both gold keyphrases, but it will also result in many other less useful keyphrases like "in my".

In the given example, the keyphrase list consists of nine tokens (lemmas, resp.) but covers only one gold keyphrase (i.e. "dog"). Noun chunks and named entities are alternative keyphrases, limiting the set of keyphrases further. Experiments where noun chunks are selected as keyphrases perform best for this example. Named entities are too restrictive, but applicable for identifying relevant entities in a text. This is useful for tasks that are targeted towards entities, e.g. for finding experts (Dörner et al., 2007) in a collection of domain-dependent texts. The selection of a linguistic type is not limited as preprocessing components might introduce further types.

2.3 Filtering

Filtering can be used together with over-generating selection approaches like taking all n-grams to decrease the number of keyphrases before ranking. One possible approach is based on POS patterns. For example, using the POS patterns, `Adjective-Noun`, `Adjective`, and `Noun` limits the set of possible keyphrases to "dog", "old cat", "cat", and "garden" in the previous example. This step can also been performed as part of the selection step, however, keeping it separated enables researchers to apply filters to keyphrases of any linguistic type. DKPro Keyphrases provides the possibility to use controlled-vocabulary keyphrase extraction by filtering out all keyphrases which are not included in a keyphrase list.

Developers of keyphrase extraction approaches can create their own filter simply by extending from a base class and adding filter-specific code. Additionally, DKPro Keyphrases does not impose workflow-specific requirements, such as a fixed number of filters. This leaves room for keyphrase extraction experiments testing new or extended filters.

2.4 Ranking

In this step, a ranker assigns a score to each remaining keyphrase candidate. DKPro Keyphrases contains rankers based on the candidate position, frequency, tf-idf, TextRank (Mihalcea and Tarau, 2004), and LexRank (Erkan and Radev, 2004).

DKPro Keyphrases also contains a special extension of tf-idf, called tf-idf$_{web}$, for which Google web1t (Brants and Franz, 2006) is used for obtaining approximate df counts. In case of keyphrase extraction for a single document or for domain-independent keyphrase extraction, web1t provides reliable n-gram statistics without any domain-dependence.

2.5 Evaluation

DKPro Keyphrases ships with all the metrics that have been traditionally used for evaluating keyphrase extraction. Kim et al. (2010) use precision and recall for a different number of keyphrases (5, 10 and 15 keyphrases). These metrics are widely used for evaluation in information retrieval. Precision @5 is the ratio of true positives in the set of extracted keyphrases when 5 keyphrases are extracted. Recall @5 is the ratio of true positives in the set of gold keyphrases when 5 keyphrases are extracted. Moreover, DKPro Keyphrases evaluates with MAP and R-precision. MAP is the mean average precision of extracted keyphrases from the highest scored keyphrase to the total number of extracted keyphrases. For each position in the rank, the precision at that position will be computed. Summing up the precision at each recall point and then taking its average will return the average precision for the text being evaluated. The mean average precision will be the mean from the sum of each text's average precision from the dataset. R-precision is the ratio of true positives in the set of extracted keyphrases, when the set is limited to the same size as the set of gold keyphrases (Zesch and Gurevych, 2009).

3 Experimental framework

In this section, we show how researchers can perform experiments covering many different configurations for preprocessing, selection, and ranking. To facilitate the construction of experiments, the framework contains a module to make its architecture compatible to the DKPro Lab framework (Eckart de Castilho and Gurevych, 2011), thus allowing to sweep through the *parameter space* of configurations. The parameter space is the combination of all possible parameters, e.g. one parameter with two possible values for preprocessing and a second parameter with two values for rankers lead to four possible combinations. We refer to *parameter sweeping experiments* when running the experiment with all possible combinations.

DKPro Keyphrases divides the experimental setup in three *tasks*. Tasks are processing steps defined in the Lab framework, which – in case of keyphrase extraction – are based on the steps described in Section 2. In the first task, the input text is fed into a pipeline and preprocessed. In the second task, the keyphrases are selected and filtered. In the third and final task they are ranked and evaluated. The output of the first two tasks are serialized objects which can be processed further by the following task. The output of the third task is a report containing all configurations and results in terms of all evaluation metrics.

The division into three tasks speeds up processing of the entire experiment. Each task has multiple configuration parameters which influence the forthcoming tasks. Instead of running the preprocessing tasks for every single possible combination, the intermediate objects are stored once and then used for every possible configuration in the keyphrase selection step.

To illustrate the advantages of experimental settings in DKPro Keyphrases, we run the previously used example sentence through the entire parameter space. Hence, tokens, lemmas, n-grams, noun chunks, and named entities will be combined with all filters and all rankers (not yet considering all possible parameters). This results in more than 10,000 configurations. Although the number of configurations is high, the computation time is low[2] as not the entire pipeline needs to run that often. This scales well for longer texts.

The experimental framework runs all possible combinations automatically and collects individual results in a report, such as a spreadsheet or text file. This allows for comparing results of different rankers, mitigating the influence of different preprocessing and filtering components. This way, the optimal experimental configuration can be found empirically. It is a great improvement for researchers because a variety of system configurations can be compared without the effort of reimplementing the entire pipeline.

Code example 1 shows the main method of an example experiment, selecting all tokens as possible keyphrases and ranking them with their tf-idf values. Lines 1 to 34 show values for dimensions which span the parameter space. A dimension consists of an identifier, followed by one or more values. Lines 36 to 40 show the creation of tasks, and in lines 42 to 48 the tasks and a report are added to one *batch task*, which is then executed. Researchers can run multiple configurations by setting multiple values to a dimension. Line 25 shows an example of a dimension with two values (using the logarithm or unchanged text frequency), in this case two configurations[3] for the ranker based on tf-idf scores.

Code example 1: Example experiment

```
1  ParameterSpace params = new
       ParameterSpace(
2   Dimension.create("language", "en"),
3   Dimension.create("frequencies",
       "web1t"),
4   Dimension.create("tfidfFeaturePath",
       Token.class"),
5
6   Dimension.create("dataset",
       datasetPath),
7   Dimension.create("goldSuffix", ".key"),
8
9  //Selection
10  Dimension.create("segmenter",
       OpenNlpSegmenter.class),
11  Dimension.create("keyphraseFeaturePath",
       Token.class),
12
13 //PosSequence filter
14  Dimension.create("runPosSequenceFilter",
       true),
15  Dimension.create("posSequence",
       standard),
16
17 //Stopword filter
18  Dimension.create("runStopwordFilter",
       true),
19  Dimension.create("stopwordlists",
       "stopwords.txt"),
20
21 // Ranking
```

[2]Less than five minutes on a desktop computer with a 3.4 GHz 8-core processor.

[3]DKPro Keyphrases provides ways to configure experiments using Groovy and JSON.

```
 Dimension.create("rankerClass",
     TfidfRanking.class),

//TfIdf
 Dimension.create("weightingModeTf",
     NORMAL, LOG),
 Dimension.create("weightingModeIdf",
     LOG),
 Dimension.create("tfidfAggregate",
     MAX),

//Evaluator
 Dimension.create("evalMatchingType",
     MatchingType.Exact),
 Dimension.create("evalN", 50),
 Dimension.create("evalLowercase",
     true),
 Dimension.create("evalType",
     EvaluatorType.Lemma),
);

Task preprocessingTask = new
    PreprocessingTask();
Task filteringTask = new
    KeyphraseFilteringTask();
candidateSelectionTask.addImport(
    preprocessingTask,
    PreprocessingTask.OUTPUT,
    KeyphraseFilteringTask.INPUT);
Task keyphraseRankingTask = new
    KeyphraseRankingTask();
keyphraseRankingTask.addImport(
    filteringTask,
    KeyphraseFilteringTask.OUTPUT,
    KeyphraseRankingTask.INPUT);

BatchTask batch = new BatchTask();
batch.setParameterSpace(params);
batch.addTask(preprocessingTask);
batch.addTask(candidateSelectionTask);
batch.addTask(keyphraseRankingTask);
batch.addReport(
    KeyphraseExtractionReport.class);
Lab.getInstance().run(batch);
```

A use case for the experimental framework is the evaluation of new preprocessing components. For example, keyphrase extraction should be evaluated with Twitter data: One collects a dataset with tweets and their corresponding keyphrases (possibly, the hash tags). The standard preprocessing will most likely fail as non-canonical language will be hard to process (e.g. *hash tags* or emoticons).

The preprocessing components can be set as a parameter and compared directly without changing the remaining parameters for filters and rankers. This allows researchers to perform reliable extrinsic evaluation of their components in a keyphrase extraction setting.

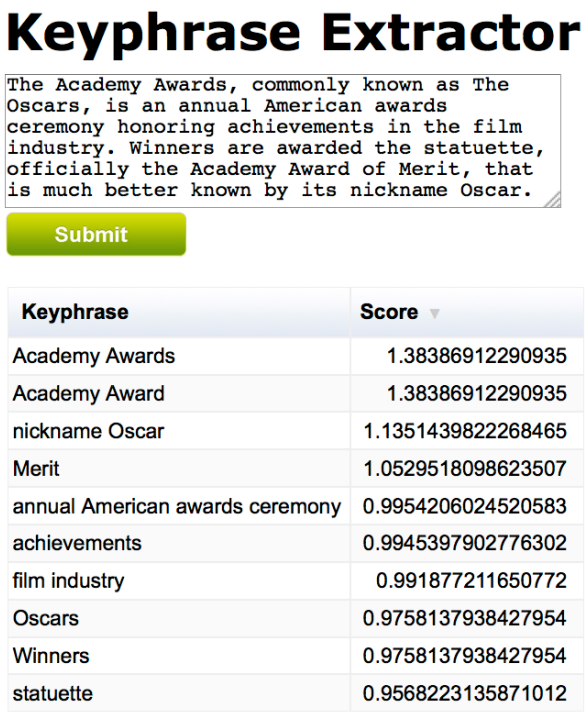

Figure 2: Screenshot of web demo in DKPro Keyphrases

4 Visualization and wrappers

To foster analysis of keyphrase extraction experiments, we created a web-based visualization framework with Spring[4]. It allows for running off-the-shelf experiments and manually inspecting results without the need to install any additional software. Figure 2 shows a visualization of one preconfigured experiment. The web demo is available online.[5] Currently, a table overview of extracted keyphrases is implemented, but developers can change it to highlighting all keyphrases. The latter is recommend for a binary classification of keyphrases. This is the case, if a system only returns keyphrases with a score above a certain threshold. The table in Figure 2 shows keyphrases with the assigned scores, which can be sorted to get a ranking of keyphrases. However, the visualization framework does not provide any evaluation capabilities.

To help new users of DKPro Keyphrases, it includes a module with two demo experiments using preconfigured parameter sets. This is especially useful for applying keyphrase extraction in other tasks, e.g. text summarization (Goldstein et

[4]http://projects.spring.io/spring-ws/

[5]https://dkpro.ukp.informatik.tu-darmstadt.de/DKProWebDemo/livedemo/3

al., 2000). Both demo experiments are frequently used keyphrase extraction systems. The first one is based on TextRank (Mihalcea and Tarau, 2004) and the second one is based on the supervised system KEA (Witten et al., 1999). Both configurations do not require any additional installation of software packages.

This module offers setters to configure parameters, e.g. the size of co-occurrence windows in case of the TextRank extractor.

5 Related work

Most work on keyphrase extraction is not accompanied with free and open software. The tools listed in this section allow users to combine different configurations with respect to preprocessing, keyphrase selection, filtering, and ranking. In the following, we give an overview of software tools for keyphrase extraction.

KEA (Witten et al., 1999) provides a Java API, which offers automatic keyphrase extraction from texts. They provide a supervised approach for keyphrase extraction. For each keyphrase, KEA computes frequency, position, and semantic relatedness as features. Thus, for using KEA, the user needs to provide annotated training data. KEA generates keyphrases from n-grams with length from 1 to 3 tokens. A controlled vocabulary can be used to filter keyphrases. The configuration for keyphrase selection and filtering is limited compared to DKPro Keyphrases, which offers capabilities for changing the entire preprocessing or adding filters.

Maui (Medelyan et al., 2009) enhances KEA by allowing the computation of semantic relatedness of keyphrases. It uses Wikipedia as a thesaurus and computes the keyphraseness of each keyphrase, which is the number of times a candidate was used as keyphrase in the training data (Medelyan et al., 2009).

Although Maui provides training data along with their software, this training data is highly domain-specific. A shortcoming of KEA and Maui is the lack of any evaluation capabilities or the possibility to run parameter sweeping experiments. DKPro Keyphrases provides evaluation tools for automatic testing of many parameter settings.

Besides KEA and Mau, which are Java systems, there are several modules in Python, e.g. topia.termextract[6], which offer capabilities for tokenization, part-of-speech tagging and keyphrase extraction. Keyphrase extraction from topia.termextract is based on noun phrases and ranks them according to their frequencies.

BibClassify[7] is a python module which automatically extracts keywords from a text based on the occurrence of terms in a thesaurus. The ranker is frequency-based like topia.termextract. BibClassify and topia.termextract do not provide evaluation capabilities or parameter sweeping experiments.

Besides these software tools, there exist web services for keyphrase extraction. AlchemyAPI[8] offers a web service for keyword extraction. It may return keyphrases encoded in various markup languages. TerMine[9] offers a SOAP service for extracting keyphrases from documents and a web demo. The input must be a String and the extracted terms will be returned as a String. Although web services can be integrated easily due to their protocol stacks, they are not extensible and replicability cannot be guaranteed over time.

6 Conclusions and future work

We presented DKPro Keyphrases, a framework for flexible and reusable keyphrase extraction experiments. This helps researchers to effectively develop new keyphrase extraction components without the need to re-implement state-of-the-art approaches.

The UIMA-based architecture of DKPro Keyphrases allows users to easily evaluate keyphrase extraction configurations. Researchers can integrate keyphrase extraction with different existing linguistic preprocessing components offered by the open-source community and evaluate them in terms of all commonly used evaluation metrics.

As future work, we plan to wrap further third-party libraries with keyphrase extraction approaches in DKPro Keyphrases and to add a supervised system using the unsupervised components as features. We expect that a supervised system using a large variety of features would improve the state of the art in keyphrase extraction.

[6] https://pypi.python.org/pypi/topia.termextract/

[7] http://invenio-demo.cern.ch/help/admin/bibclassify-admin-guide

[8] http://www.alchemyapi.com/api/keyword-extraction/

[9] http://www.nactem.ac.uk/software/termine/

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806, by the Klaus Tschira Foundation under project No. 00.133.2008, and by the German Federal Ministry of Education and Research (BMBF) within the context of the Software Campus project *open window* under grant No. 01IS12054. The authors assume responsibility for the content. We thank Richard Eckart de Castilho and all contributors for their valuable collaboration and the we thank the anonymous reviewers for their helpful comments.

References

Thorsten Brants and Alex Franz. 2006. Web 1T 5-Gram Corpus Version 1.1. Technical report, Google Research.

Christian Dörner, Volkmar Pipek, and Markus Won. 2007. Supporting Expertise Awareness: Finding Out What Others Know. In *Proceedings of the 2007 Symposium on Computer Human Interaction for the Management of Information Technology*.

Richard Eckart de Castilho and Iryna Gurevych. 2009. DKPro-UGD: A Flexible Data-Cleansing Approach to Processing User-Generated Discourse. In *Online-proceedings of the First French-speaking meeting around the framework Apache UIMA*.

Richard Eckart de Castilho and Iryna Gurevych. 2011. A Lightweight Framework for Reproducible Parameter Sweeping in Information Retrieval. In *Proceedings of the 2011 Workshop on Data Infrastructures for Supporting Information Retrieval Evaluation*, pages 7–10.

Günes Erkan and Dragomir Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

David Ferrucci and Adam Lally. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10(3-4):327–348.

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-Document Summarization By Sentence Extraction. In *Proceedings of the NAACL-ANLP 2000 Workshop: Automatic Summarization*, pages 40–48.

Iryna Gurevych, Max Mühlhäuser, Christof Müller, Jürgen Steimle, Markus Weimer, and Torsten Zesch. 2007. Darmstadt Knowledge Processing Repository Based on UIMA. In *Proceedings of the First Workshop on Unstructured Information Management Architecture at Biannual Conference of the Society for Computational Linguistics and Language Technology*.

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.

Cody Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling Question Answering to the Web. *ACM Transactions on Information Systems*, 19(3):242–262.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *An Introduction to Information Retrieval*. Cambridge University Press Cambridge.

Olena Medelyan, Eibe Frank, and Ian H Witten. 2009. Human-competitive Tagging using Automatic Keyphrase Extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1318–1327.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 404–411.

Min Song, Il Yeol Song, Robert B. Allen, and Zoran Obradovic. 2006. Keyphrase Extraction-based Query Expansion in Digital Libraries. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 202–209.

Simon Tucker and Steve Whittaker. 2009. Have A Say Over What You See: Evaluating Interactive Compression Techniques. In *Proceedings of the 2009 International Conference on Intelligent User Interfaces*, pages 37–46.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Andrew Gutwin, and Craig G . Nevill-Manning. 1999. KEA: Practical Automatic Keyphrase Extraction. In *Proceedings of the 4th ACM Conference on Digital Libraries*, pages 254–255.

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.