pke: an open source python-based keyphrase extraction toolkit

Florian Boudin

pke: an open source python-based keyphrase extraction toolkit

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

Abstract

We describe pke, an open source python-based keyphrase extraction toolkit. It provides an end-to-end keyphrase extraction pipeline in which each component can be easily modified or extented to develop new approaches. pke also allows for easy benchmarking of state-of-the-art keyphrase extraction approaches, and ships with supervised models trained on the SemEval-2010 dataset (Kim et al., 2010).

1 Introduction

Keyphrase extraction is the task of identifying the words and phrases that represent the main topics of a document. Keyphrases have been shown to be useful for a variety of natural language processing applications such as document indexing (Gutwin et al., 1999), text categorization (Hulth and Megyesi, 2006) or summarization (Qazvinian et al., 2010). Recent years have witnessed increased interest in keyphrase extraction (Gollapalli et al., 2015), and several benchmark datasets have become available in various domains and languages (Hasan and Ng, 2014). Yet, there are few tools available for automatic keyphrase extraction, and none of them offer implementations of current state-of-the-art approaches nor the suitability for rapid prototyping like the python-based Natural Language Toolkit (nltk) (Bird et al., 2009) does. In this demonstration, we describe an open source python-based keyphrase extraction toolkit, called pke, which 1) provides implementations of existing supervised and unsupervised keyphrase extraction approaches; 2) can be easily extended to develop new approaches; 3) ships with a collection of already trained models, which are ready for use. The pke toolkit is open source under the GNU GPL licence and available at https://github.com/boudinfl/pke

2 Architecture

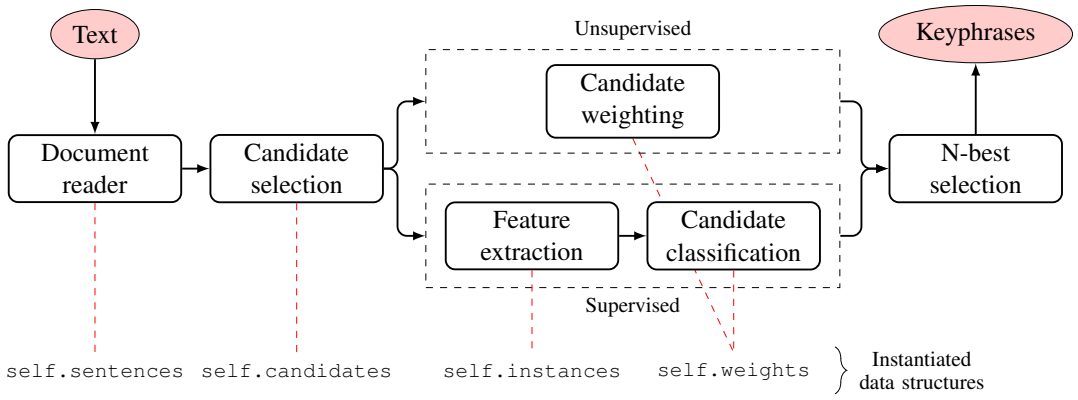

Figure 1: Overall architecture of pke.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations, pages 69–73, Osaka, Japan, December 11-17 2016.

The overall architecture of `pke` is depicted in Figure 1. Extracting keyphrases from an input document involves three steps. First, keyphrase candidates (i.e. words and phrases that are eligible to be keyphrases) are selected from the content of the document. Second, candidates are either ranked using a candidate weighting function (unsupervised approaches), or classified as keyphrase or not using a set of extracted features (supervised approaches). Third, the top-N highest weighted candidates, or those classified as keyphrase with the highest confidence scores, are selected as keyphrases.

Document reader: three input formats are supported: raw text, preprocessed text[1] and Stanford CoreNLP XML (Manning et al., 2014). When raw text is provided, preprocessing (i.e. tokenization, sentence splitting and POS-tagging) is carried out using `nltk`. Preprocessed text files are expected to use POS tags from the Penn Treebank tagset. Document logical structure information[2], used as features in some supervised approaches, can by specified by incorporating attributes into the sentence elements of the CoreNLP XML format.

Implemented approaches: The `pke` toolkit currently implements the following approaches, each consisting of a unique combination of candidate selection and candidate ranking methods.

Unsupervised	`TfIdf`: we re-implemented the TF×IDF n-gram based baseline in (Kim et al., 2010). By default, it uses 1, 2, 3-grams as keyphrase candidates and filter out those shorter than 3 characters, containing words made of only punctuation marks or one character long[3]. `SingleRank` (Wan and Xiao, 2008): keyphrase candidates are the sequences of adjacent nouns and adjectives. Candidates are ranked by the sum of their words scores, computed using TextRank (Mihalcea and Tarau, 2004) on a word-based graph representation of the document. `TopicRank` (Bougouin et al., 2013): this model improves `SingleRank` by grouping lexically similar candidates into topics and directly ranking topics. Keyphrases are produced by extracting the first occurring candidate of the highest ranked topics. `KP-Miner` (El-Beltagy and Rafea, 2010): keyphrase candidates are sequences of words that do not contain punctuation marks or stopwords[4]. Candidates that appear less than three times or that first occur beyond a certain position are removed. Candidates are then weighted using a modified TF×IDF formula that account for document length.
Supervised	`Kea` (Witten et al., 1999): keyphrase candidates are 1, 2, 3-grams that do not begin or end with a stopword. Keyphrases are selected using a naïve bayes classifier with two features: TF×IDF and the relative position of first occurrence. `WINGNUS` (Nguyen and Luong, 2010): keyphrase candidates are simplex nouns and noun phrases detected using a set of POS filtering rules. Keyphrases are then selected using a naïve bayes classifier with a large set of features including document logical structure information.

Already trained models: to promote benchmarking of current state-of-the-art keyphrase extraction approaches on new datasets, we make available supervised models for `Kea` and `WINGNUS`, as well as document frequency counts, trained on the training part of the SemEval-2010 dataset (Kim et al., 2010).

Non English languages: while the default language in `pke` is English, extracting keyphrases from documents in other languages is easily achieved by inputting already preprocessed documents, and setting the `language` parameter to the desired language. The only language dependent resources used in `pke` are the stoplist and the stemming algorithm from `nltk` that are available in 11 languages[5]. Examples of use for other languages are provided in the documentation.

[1] whitespace-separated POS-tagged tokens, one sentence per line.
[2] We use the classification categories proposed by Luong et al. (2012).
[3] This filtering process is also applied to the other models.
[4] We use the stoplist in nltk, `http://www.nltk.org`
[5] `http://www.nltk.org/_modules/nltk/corpus.html`

3 Elementary Usage

Python Library: `pke` can be imported as a Python module, which is its primary use. Figure 2 gives a complete example of use, showing the typical three-step process involved in keyphrase extraction. Particular attention was paid to modularity: each method instantiates a different data structure (see Figure 1), thus making it easier to develop new approaches by modifying the behaviour of only some components. Modifying the example to apply another approach is quite straightforward: replace `TopicRank` at line 4 with another model (e.g. `TfIdf`).

```
1  import pke
2
3  # initialize TopicRank
4  extr = pke.TopicRank(input_file='/path/to/input')
5
6  # load the content of the document
7  extr.read_document(format='raw')
8
9  # step 1: candidate selection
10 extr.candidate_selection()
11
12 # step 2: candidate weighting
13 extr.candidate_weighting()
14
15 # step 3: N-best selection
16 keyphrases = extr.get_n_best(n=10)
```

Figure 2: Example of keyphrase extraction using `TopicRank` with `pke`.

Figure 3 illustrates how to train a new supervised model in `pke`. The training data consists of a set of documents along with a reference file containing annotated keyphrases in the SemEval-2010 format[6]. Candidate classification is performed using the implementations available in scikit-learn [7].

```
1  import pke
2
3  # load document frequency counts (DF) as a dictionary
4  df_counts = pke.load_document_frequency_file('/path/to/file')
5
6  # train new Kea model
7  pke.train_supervised_model(input_dir='/path/to/input/directory/',
8                             reference_file='/path/to/reference/file',
9                             model_file='/path/to/model/file',
10                            df=df_counts,
11                            model=pke.Kea())
```

Figure 3: Training a new `Kea` supervised model with `pke`.

Command Line: the `pke` toolkit also includes a command line tool that allows users to perform keyphrase extraction without any knowledge of the Python programming language. An example of use is given below.

```
python cmd_pke.py -i /path/to/input -f raw -o /path/to/output -a TopicRank
```

Here, unsupervised keyphrase extraction using TopicRank is performed on a raw text input file, and the top ranked keyphrase candidates are outputted into a file.

4 Benchmarking

We evaluate the performance of our re-implementations using the SemEval-2010 benchmark dataset (Kim et al., 2010). This dataset is composed of 244 scientific articles (144 in training and 100

[6] `http://docs.google.com/Doc?id=ddshp584_46gqkkjng4`
[7] `http://scikit-learn.org`

for test) collected from the ACM Digital Library (conference and workshop papers). Document logical structure information, required to compute features in the WINGNUS approach, is annotated with ParsCit (Kan et al., 2010)[8]. The Stanford CoreNLP pipeline[9] (tokenization, sentence splitting and POS-tagging) is then applied to the documents from which irrelevant pieces of text (e.g. tables, equations, footnotes) were filtered out[10].

We follow the evaluation procedure used in the SemEval-2010 competition and evaluate the performance of each implemented approach in terms of precision (P), recall (R) and f-measure (F) at the top N keyphrases. We use the set of combined author- and reader-assigned keyphrases as reference keyphrases. Extracted and reference keyphrases are stemmed to reduce the number of mismatches. Detailed results for each approach are presented in Table 1.

Approach	P	R	F
`TfIdf`	20.0	14.1	16.4
`TopicRank`	15.6	10.8	12.6
`SingleRank`	2.2	1.5	1.8
`KP-Miner`	24.1	17.0	19.8
`Kea`	23.5	16.6	19.3
`WINGNUS`	24.7	17.3	20.2

Table 1: Performance of each approach computed at the top 10 extracted keyphrases. Results are expressed as a percentage of precision (P), recall (R) and f-measure (F).

5 Related Work

Most of the tools available for automatic keyphrase extraction only implement one approach, and are often outdated with respect to the current state-of-the-art. These tools also rely on in-house text preprocessing and candidate selection/filtering pipelines, which makes it difficult to compare results across several approaches. One notable exception to this is the DKPro Keyphrases Java framework (Erbs et al., 2014), which provides a UIMA-based workbench for developing and evaluating new keyphrase extraction approaches. However, this framework requires users to learn UIMA before they can get started, and does not provide supervised approaches that are known to perform better (Hasan and Ng, 2014).

6 Conclusion

We presented `pke`, an open source python-based keyphrase extraction toolkit that provides an end-to-end pipeline in which each component can be easily modified to develop new models. `pke` includes implementations of state-of-the-art supervised and unsupervised approaches, and comes with a collection of already trained models. It is our hope that this toolkit will help researchers to compare, build upon and devise keyphrase extraction approaches.

Acknowledgments

This work was partially supported by the TALIAS project (grant of CNRS PEPS INS2I 2016, `https://boudinfl.github.io/talias/`). We thank the anonymous reviewers for their comments.

References

[Bird et al.2009] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. O'Reilly.

[8] We use ParsCit v110505.

[9] Use use Stanford CoreNLP v3.6.0.

[10] Further details about preprocessing can be found at `https://github.com/boudinfl/semeval-2010-pre`

[Bougouin et al.2013] Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of IJCNLP*, pages 543–551.

[El-Beltagy and Rafea2010] Samhaa R. El-Beltagy and Ahmed Rafea. 2010. Kp-miner: Participation in semeval-2. In *Proceedings of SemEval*, pages 190–193.

[Erbs et al.2014] Nicolai Erbs, Pedro Bispo Santos, Iryna Gurevych, and Torsten Zesch. 2014. Dkpro keyphrases: Flexible and reusable keyphrase extraction experiments. In *Proceedings of ACL*, pages 31–36.

[Gollapalli et al.2015] Sujatha Das Gollapalli, Cornelia Caragea, Xiaoli Li, and C. Lee Giles, editors. 2015. *Proceedings of the ACL 2015 Workshop on Novel Computational Approaches to Keyphrase Extraction*.

[Gutwin et al.1999] Carl Gutwin, Gordon Paynter, Ian Witten, Craig Nevill Manning, and Eibe Frank. 1999. Improving Browsing in Digital Libraries with Keyphrase Indexes. *Decision Support Systems*, 27(1):81–104.

[Hasan and Ng2014] Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of ACL*, pages 1262–1273.

[Hulth and Megyesi2006] Anette Hulth and Beáta B. Megyesi. 2006. A study on automatically extracted keywords in text categorization. In *Proceedings of COLING/ACL*, pages 537–544.

[Kan et al.2010] Min-Yen Kan, Minh-Thang Luong, and Thuy Dung Nguyen. 2010. Logical structure recovery in scholarly articles with rich document features. *Int. J. Digit. Library Syst.*, 1(4):1–23.

[Kim et al.2010] 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 SemEval*, pages 21–26.

[Luong et al.2012] Minh-Thang Luong, Thuy Dung Nguyen, and Min-Yen Kan. 2012. Logical structure recovery in scholarly articles with rich document features. *Multimedia Storage and Retrieval Innovations for Digital Library Systems*, 270.

[Manning et al.2014] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of ACL*, pages 55–60.

[Mihalcea and Tarau2004] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, pages 404–411.

[Nguyen and Luong2010] Thuy Dung Nguyen and Minh-Thang Luong. 2010. Wingnus: Keyphrase extraction utilizing document logical structure. In *Proceedings of SemEval*, pages 166–169.

[Qazvinian et al.2010] Vahed Qazvinian, Dragomir R. Radev, and Arzucan Ozgur. 2010. Citation summarization through keyphrase extraction. In *Proceedings of COLING*, pages 895–903.

[Wan and Xiao2008] Xiaojun Wan and Jianguo Xiao. 2008. Collabrank: Towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of COLING*, pages 969–976.

[Witten et al.1999] 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.