Unsupervised Keyphrase Extraction with PatternRank

Using pretrained transformer language models and part of speech for state-of-the-art keyphrase extraction

Tim Schopf · Follow

Published in Towards Data Science · 6 min read · Oct 19, 2022

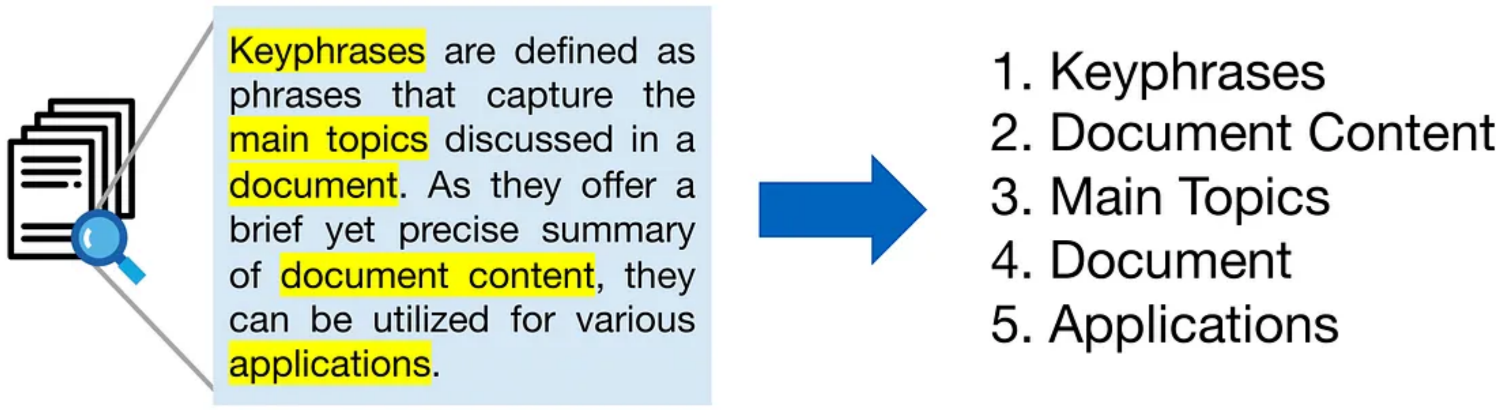

Keyphrases are defined as phrases that capture the main topics discussed in a document. As they offer a brief yet precise summary of document content, they can be utilized for various applications.
1. Keyphrases
2. Document Content
3. Main Topics
4. Document
5. Applications

Image by Author.

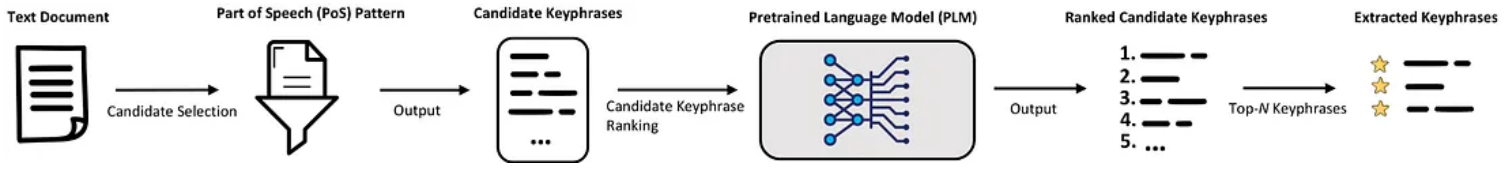
This post is based on our paper "PatternRank: Leveraging Pretrained Language Models and Part of Speech for Unsupervised Keyphrase Extraction (2022)" accepted to KDIR22. You can read more details about our approach there.

📑 To quickly get an overview of the content of a text, we can use keyphrases that concisely reflect its semantic context. Keyphrases describe the most essential aspect of a text. Unlike simple keywords, keyphrases do not consist solely of single words, but of several compound words. E.g. *"football"* vs. *"youth football training"*. Keyphrases provide a more accurate description than simple keywords and are therefore often the preferred does not rely on labeled data choice.

In this post, we present **PatternRank**, which leverages pretrained language models and part-of-speech for unsupervised keyphrase extraction from single documents. Our approach does not rely on labeled data and therefore can be easily used in a variety of different domains. Our experiments show PatternRank achieves higher precision, recall and F1-scores than previous state-of-the-art approaches 🏆. In addition, we present the KeyphraseVectorizers Python package 🐍, which allows easy modification of part-of-speech patterns for candidate keyphrase selection, and hence adaptation of our approach to any domain.

How does PatternRank work?

The figure below illustrates the general keyphrase extraction process of PatternRank👇.


Text Document
Candidate Selection
Part of Speech (PoS) Pattern
Output
Candidate Keyphrases
Candidate Keyphrase Ranking
Pretrained Language Model (PLM)
Output
Ranked Candidate Keyphrases
Top-N Keyphrases
Extracted Keyphrases

PatternRank approach for unsupervised keyphrase extraction. A single text document is used as input for an initial filtering step where candidate keyphrases are selected which match a defined part-of-speech pattern. Subsequently, the candidate keyphrases are ranked by a pretrained language model based on their semantic similarity to the input text document. Finally, the top-N keyphrases are extracted as a concise reflection of the input text document. Source: Schopf et al.

To extract keyphrases from text*, PatternRank performs the following steps:*

1. The input consists of a single text document which is being word tokenized.
2. The word tokens are then tagged with part-of-speech tags.
3. Tokens whose tags match a previously defined part-of-speech pattern are selected as candidate keyphrases.
4. Then, a pretrained language model embeds the entire text document as well as all candidate keyphrases as semantic vector representations.

5. Subsequently, the cosine similarities between the document representation and the candidate keyphrase representations are computed and the candidate keyphrases are ranked in descending order based on the computed similarity scores.
6. Finally, the top-N ranked keyphrases, which are most representative of the input document, are extracted.

In our experiments, we use the pretrained all-mpnet-base-v2 language model. It is a SBERT model that has been shown to produce state of the art text representations for semantic similarity tasks. For the scope of this blog post, we only use simple noun phrases as part-of-speech patterns. Although we show in our paper that more complex part-of-speech patterns can lead to even better results, using noun phrases in PatternRank is a simple but effective approach that can extract meaningful keyphrases in any domain🏅.

> *A **noun** is a word that usually refers to a thing or a person. An **adjective** is a word that describes a noun. A **noun phrase** is a simple phrase built around a noun. It consists of **zero or more adjectives followed by one or more nouns**. For example: a huge tree, some colourful sweets, the large, royal castle.*

Comparison of PatternRank with previous approaches

Previous keyphrase extraction approaches

Most popular unsupervised keyphrase extraction approaches can be characterized as either statistics-based, graph-based, or embedding-based methods, while *Tf-Idf* is a common baseline used for evaluation. In this post, we evaluate PatternRank against three very popular keyphrase extraction approaches. If you are interested in a detailed comparison of more keyphrase extraction approaches, check out the paper of Papagiannopoulou and Tsoumakas, (2019).

👉 YAKE is a fast and lightweight approach for unsupervised keyphrase extraction from single documents based on statistical features.

👉 SingleRank applies a ranking algorithm to word co-occurrence graphs for unsupervised keyphrase extraction from single documents.

👉 KeyBERT uses, similar to PatternRank, a pretrained language model to rank candidate keyphrases. However, KeyBERT uses simple word n-grams as candidate keyphrases rather than word tokens that match a certain part-of-speech pattern, as in our PatternRank approach.

> A word n-gram range lets users decide the length of the sequence of consecutive words that should be extracted from a given text. Let's suppose we define a `word n-gram range = (1,3)`. Then we woul*d choose to extract the unigrams (only single word), bigrams (group of two consecutive words), and the trigrams (group of three consecutive words) from t*he text. Applying the word n-gram range to `"an apple a day keeps the doctor away"` will result in `["an", "apple", "a","day", "keeps", "the", "doctor", "away", "an apple", "apple a", "a day", "day keeps", "keeps the", "the doctor", "doctor away", "an apple", "apple a day", "a day keeps", "day keeps the", "keeps the doctor", "the doctor away"]` .-**Devish Parmar**

For evaluation, we compare the performances of YAKE, SingleRank, KeyBERT and PatternRank. For KeyBERT and PatternRank, we use the same pretrained all-mpnet-base-v2 language model.

Data

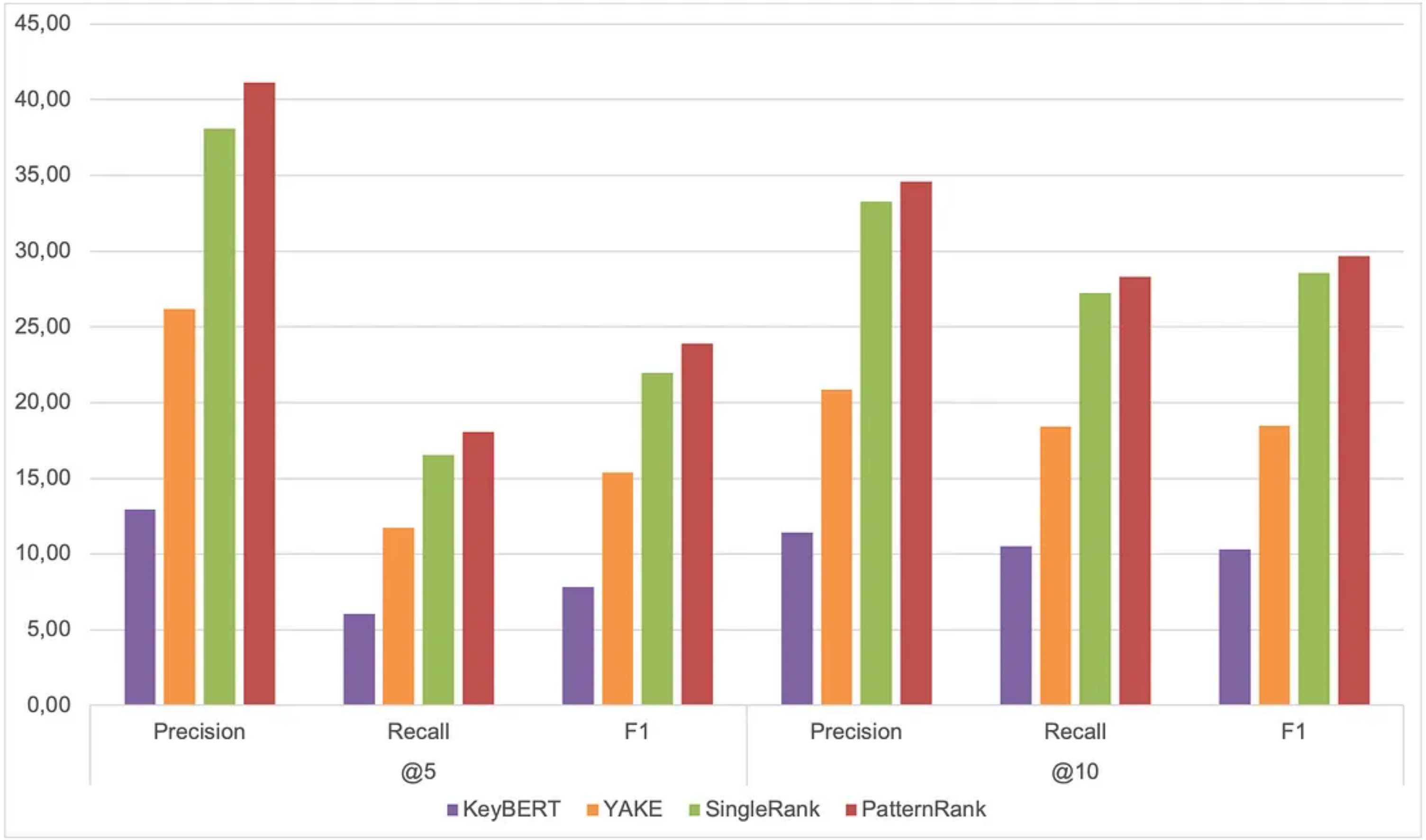
💾 We evaluate the keyphrase extraction approaches on the Inspec dataset. It consists of 2,000 English computer science abstracts collected from scientific journal articles between 1998 and 2002. Each abstract has assigned two different types of keyphrases. First, controlled and manually assigned keyphrases that appear in the thesaurus of the Inspec dataset but do not necessarily have to appear in the abstract. Second, uncontrolled keyphrases that are freely assigned by professional indexers and are not restricted to either the thesaurus or the abstract. In our experiments, we consider the union of both types of keyphrases as the gold keyphrases.

Metrics

We evaluate based on the exact match approach, which means that true positives are extracted keyphrases that have an exact string match with one of the gold keyphrases. We report *Precision@N*, *Recall@N* and *F1@N* scores, using the top-N extracted keyphrases respectively (*N=5,10*).

Results

The results of our evaluation are shown in the figure below 📊.



Comparison of KeyBERT, YAKE, SingleRank and PatternRank keyphrase extraction results on the Inspec dataset.

The results show, that PatternRank outperforms all other approaches across all benchmarks. KeyBERT uses the same pretrained language model to rank the candidate keyphrases as PatternRank, but uses simple n-grams for candidate keyphrase selection instead of part-of-speech patterns (noun phrases in this case). As a result, KeyBERT consistently performs worst among all approaches. As expected, YAKE was the fastest keyphrase extraction approach because it is a lightweight method based on statistical features. However, the extracted keyphrases are not very accurate and in comparison to PatternRank, YAKE significantly performs worse in all evaluations. SingleRank is the only approach that achieves competitive results compared to PatternRank. Nevertheless, it consistently performs a few percentage points worse than PatternRank across all evaluations. We therefore conclude that PatternRank achieves state-of-the-art keyphrase extraction results 🏆.

How to use PatternRank?

We developed the KeyphraseVectorizers Python package 🐍 which makes **PatternRank** easy to use. Using KeyphraseVectorizers together with KeyBERT for keyphrase extraction results in the PatternRank approach. The idea is to use the implementation of KeyBERT for ranking keyphrases with pretrained language models together with the ability of candidate keyphrase selection with part-of-speech patterns from KeyphraseVectorizers to yield the **PatternRank** approach. How to implement this approach in Python exactly is explained in this blog post💡.

Summary

PatternRank is a recently developed approach which can be used to extract state-of-the-art keyphrases from text documents. The evaluation shows that PatternRank performs best in terms of precision, recall and F1-score across all benchmarks. In addition, we present the KeyphraseVectorizers Python package 🐍, which makes PatternRank easy to use and customize.

Big thanks also to Maarten Grootendorst, who supported us with input and inspiration while writing the KeyphraseVectorizers package.

Sources

PatternRank: Leveraging Pretrained Language Models and Part of Speech for Unsupervised Keyphrase...

Keyphrase extraction is the process of automatically selecting a

arxiv.org

GitHub — TimSchopf/KeyphraseVectorizers: Set of vectorizers that extract keyphrases with...

Set of vectorizers that extract keyphrases with part-of-speech patterns from a collection of text documents and convert...

github.com

NLP AI Data Science Keyphrase Extraction Naturallanguageprocessing

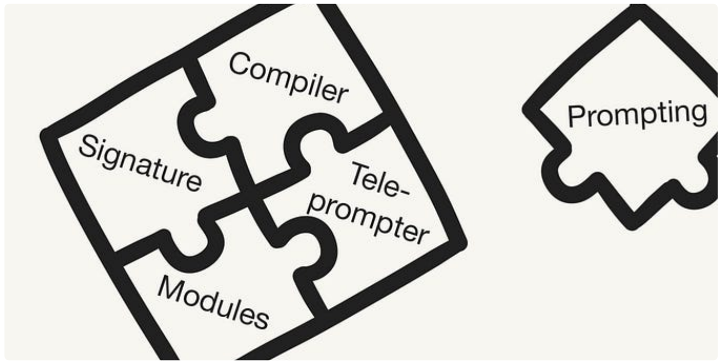
Written by Tim Schopf

Follow

361 Followers · Writer for Towards Data Science

PhD candidate @ Technical University of Munich | Passionate about NLP and Knowledge Graphs! Get in touch: https://de.linkedin.com/in/tim-schopf

More from Tim Schopf and Towards Data Science



Tim Schopf in Towards Data Science

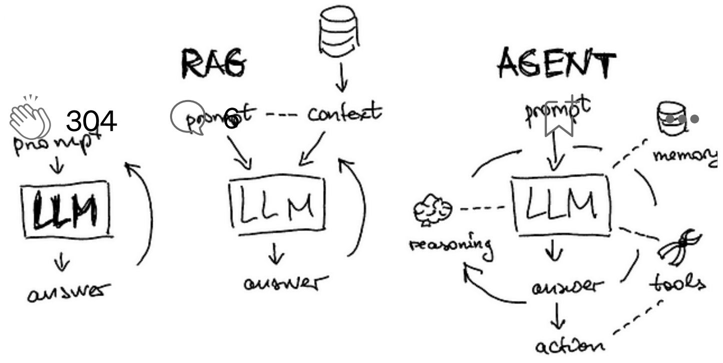
Unsupervised Text Classification with Lbl2Vec

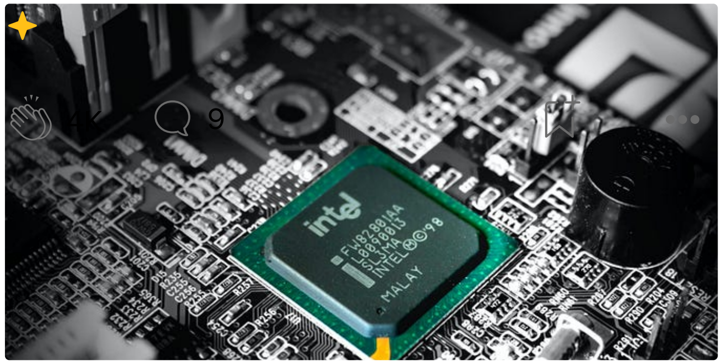
An introduction to embedding-based classification of unlabeled text documents

Leonie Monigatti in Towards Data Science

Intro to DSPy: Goodbye Prompting, Hello Programming!

How the DSPy framework solves the fragility problem in LLM-based applications by...





Alex Honchar in Towards Data Science

Intro to LLM Agents with Langchain: When RAG is Not...

First-order principles of brain structure for AI assistants

7 min read · Mar 15, 2024

1.6K 8

Tim Schopf in Towards Data Science

Parallel Inference of HuggingFace 🤗 Transformers on CPUs

An introduction to multiprocessing predictions of large machine learning and...

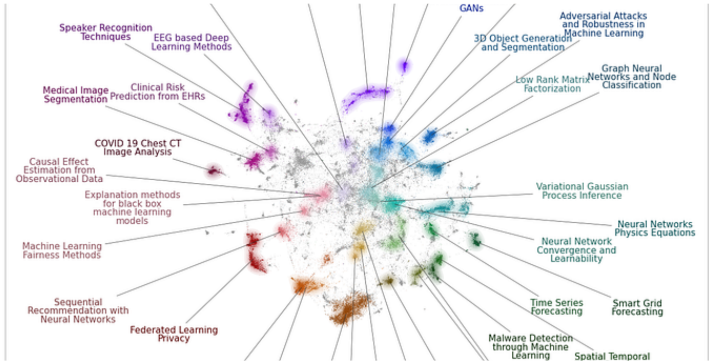
6 min read · Feb 21, 2022

215 4

See all from Tim Schopf

See all from Towards Data Science

Recommended from Medium



Ashwin Rachha

Topic Modeling with Quantized Large Language Models (LLMs): ...

· 12 min read · Jan 8, 2024

285 1

Maarten Grootendorst in Towards Data Science

Introducing KeyLLM — Keyword Extraction with LLMs

Use KeyLLM, KeyBERT, and Mistral 7B to extract keywords

· 9 min read · Oct 5, 2023

1.1K 15

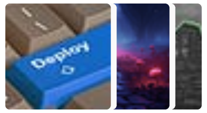
Lists

The New Chatbots: ChatGPT, Bard, and Beyond

12 stories · 350 saves

Natural Language Processing

1342 stories · 822 saves



Predictive Modeling w/ Python

20 stories · 1055 saves

ChatGPT prompts

47 stories · 1355 saves

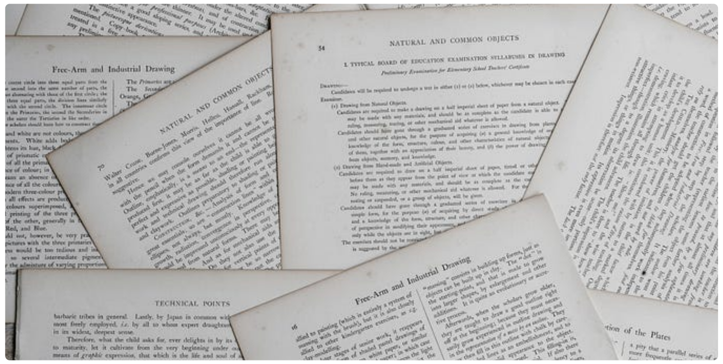
Ahmed Mellit

Text Similarity Implementation using BERT Embedding in Python

Unlocking the Power of BERT and Cosine Similarity: Enhancing Text Similarity Analysis

5 min read · Nov 9, 2023

63



Philipp Warmer

Conditional Relationship Extraction

LLMs as smart pivot tables to extract entities and their predefined interrelations from...

5 min read · Dec 5, 2023

68 1

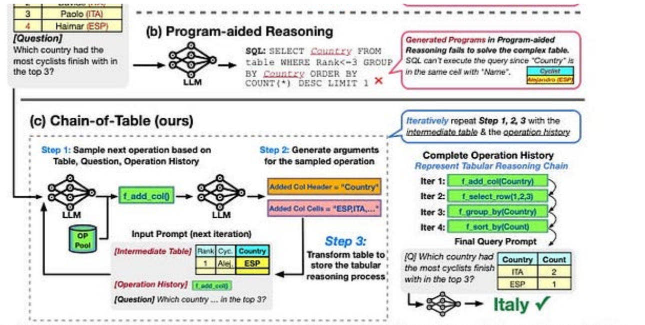

(b) Program-aided Reasoning
[Question] Which country had the most cyclists finish with in the top 3?
LLM
SQL: SELECT Country FROM table WHERE Rank<=3 GROUP BY Country ORDER BY COUNT(*) DESC LIMIT 1
Generated Programs in Program-aided Reasoning fails to solve the complex table. SQL can't execute the query since "Country" is in the same cell with "Name".
(c) Chain-of-Table (ours)
Iteratively repeat Step 1, 2, 3 with the intermediate table & the operation history
Step 1: Sample next operation based on Table, Question, Operation History
Step 2: Generate arguments for the sampled operation
f_add_col()
Added Col Header = "Country"
Added Col Cells = "ESP,ITA,..."
OP Pool
Input Prompt (next iteration)
[Intermediate Table]
[Operation History]
[Question] Which country ... in the top 3?
Step 3: Transform table to store the tabular reasoning process
Complete Operation History
Represent Tabular Reasoning Chain
Iter 1: f_add_col(Country)
Iter 2: f_select_row(1,2,3)
Iter 3: f_group_by(Country)
Iter 4: f_sort_by(Count)
Final Query Prompt
[Q] Which country had the most cyclists finish with in the top 3?
Italy ✓

Figure 1: Illustration of the comparison between (a) generic reasoning, (b) program-aided reasoning, and (c) the proposed CHAIN-OF-TABLE. Given a complex table where a cyclist's nationality and name are in the same cell,

Aditya Mohan in Towards AI

Demystifying Information Extraction using LLM

Extracting structured data out of unstructured text has never been easier!

7 min read · Jan 4, 2024

157 1

Angelina Yang

Chain-of-Table: How to Talk to Your Data Directly? 🙀

The idea of Text2SQL isn't groundbreaking: instead of crafting SQL queries yourself, you...

✦ · 3 min read · Jan 25, 2024

25

See more recommendations

Help Status About Careers Blog Privacy Terms Text to speech Teams