Application of KEA for Semantically Associated Structural Units Search in a Corpus and Text Summarization

Elena Sokolova

Saint Petersburg State University, Saint Petersburg, Russia
st049868@student.spbu.ru

Abstract. This paper presents results of the research on possible applications of keyphrase extraction algorithm KEA. Although this algorithm is widely used as an effective and universal tool for keyphrase extraction, our study is aimed at exploration of its further adjustments in the tasks of translation equivalents search and for semantic compression, namely, for extractive summarization. To be precise, in our first series of experiments we analyzed the output of KEA based on the text corpus developed from the United Nations documents in order to find semantically associated structural units (possible translation equivalents) among Russian and English keyphrases. The second series of experiments is concerned with using keyphrases automatically extracted by KEA to compose extracts for short stories. In this case we also compiled a corpus of short stories written in (or translated into) Russian and adjusted KEA so that ranked sentences with keyphrases could be used to form previews for the stories.

Keywords: keyphrase extraction, KEA, translation equivalents, summarization.

1 Introduction

Keyphrases have a wide range of practical applications in rather different fields such as document summarization, indexing, information retrieval, library systems, etc. Being structural units themselves, keyphrases convey the most important information about the content of the document. That is why automatic keyphrase extraction is one of the most highly sought tasks to solve today.

There are different approaches to extract keyphrases from a document [1, 2]: statistical (TFxIDF, Chi-square, C-value, Log-Likelihood, etc.), linguistic (including different levels of linguistic analysis), machine learning (Naïve Bayes classifier, SVM, etc.) and also hybrid algorithms (KEA).

In this paper we explore further implementations of one of commonly known keypharse extraction algorithms KEA (Keyphrase Extraction Algorithm) in the wide field of Natural Language Processing (NLP) [3]. Therefore, we conducted a series of experiments trying to adjust KEA to the tasks which combine semantic compression and text transformations.

To be precise, in the first experiment we try to find out if KEA is capable of finding semantically related unites, such as translation equivalents, synonyms, hyponyms, etc., for two different languages, namely Russian and English.

The second experiment is devoted to the possibility of using KEA as an intermediate tool for an extractive summarization [4] algorithm. Keyphrases automatically extracted by KEA were used to identify salient sentences in the text.

To mark the borders of our research, it needs to be noted that we are not trying to find new solutions to existing problems in the field of NLP. The subject of our study is KEA itself, namely, how it can be used and what for. Thus, those applications of KEA that we will consider further represent only one of all possible varieties of approaches to solving some certain tasks, and also give new information about KEA's abilities. Despite the fact that the algorithm is not precisely new, we have chosen KEA for our experiments because it proved to be a useful and universal tool in different fields, but so far has not been used for processing Russian texts.

We would also like to state in advance that, as a significant part of the research was conducted manually, in many aspects it is not large-scale.

The paper is organized as follows. Section 2 briefly describes the structure and working principles of KEA. Section 3 contains description of the first possible KEA application, namely identification of translation equivalents, while Section 4 deals with the second experiment which concerns composing extracts for short stories based on keyphrases extracted by KEA. Section 5 is devoted to general conclusions and future work.

2 KEA Structure

KEA was developed by I.H. Witten et al. in New Zealand in 1999 [5, 6]. It is a keyphrase extraction algorithm which contains two stages:

- training: KEA is trained on the documents where the keyphrases are manually assigned by the author; as a result, a model for identifying keyphrases in new documents is created;
- extraction: the model created on the previous step is applied, and the keyphrases for new documents are identified.

On both stages, by certain rules, KEA chooses candidate phrases. The procedure of candidate selection is as follows:

1) preprocessing of the input documents:
   - tokenization;
   - relative phrase boundaries are placed;
   - non-alphabetical characters are removed.
2) keyphrase candidates filtering:
   - the length of a candidate keyphrase is limited to a certain size;
   - proper names cannot be chosen as candidate keyphrases;
   - constructions beginning or ending with a stopword cannot be candidate keyphrases.
3) case-folding and stemming.

After that for each candidate two features – *TFxIDF* and *first occurrence* – are calculated. *TFxIDF* shows how often a phrase occurs in the document in comparison to its frequency in some large corpus:

$$TF \times IDF = \frac{freq(P,D)}{size(D)} \times -log_2 \frac{df(P)}{N}, \text{where}$$

freq(*P*,*D*) is the number of times *P* occurs in *D*;

size(*D*) is the number of words in *D*;

df(*P*) is the number of documents of some collection of documents or in some corpus containing *P*;

N is the size of the collection or corpus.

The second feature, *first occurrence*, is the distance between a phrase first appearance and the beginning of the document, divided by the number of words in the document. The result is a number between 0 and 1.

After being trained, KEA marks each candidate as a keyphrase or non-keyphrase, which is a class future used later by Naïve Bayes classifier. Then, by applying the model built on the training stage, KEA selects keyphrases from a new document and after some post-processing operations represents the best keyphrases to a user.

When the classifier processes a candidate phrase with feature values *t* (*TF×IDF*) and *d* (*distance*), two quantities are calculated:

$$P[yes] = \frac{Y}{Y+N} P_{\text{TF×IDF}}[t|yes] P_{distance}[d|yes]$$

and the same for *P*[*no*], where *Y* is the number of positive instances in the training set, i.e. keyphrases assigned by the author, and *N* is the number of negative instances, i.e. candidate phrases which are not keyphrases.

The overall probability that a candidate phrase is a keyphrase, in its turn, is calculated in the following way:

$$p = P[yes]/(P[yes] + P[no])$$

According to this value, candidate keypharses are ranked and the first *r*, where *r* is a requested number of keyphrases, presented to the user.

3 Translation equivalents among Russian and English keyphrases automatically extracted by KEA

3.1 Collecting and preprocessing text corpora

Besides KEA's possible practical usages this experiment was also aimed at verifying, to which extend KEA is a language independent tool. For us it would mean that it is capable to identify conventionally 'the same words' for the same document written in several languages. For this purpose we developed a corpus using the United Nations (the UN) documents [7] as official papers have at most precise translation and are written in formal style.

The corpus contains official letters, declarations, protocols, reports, etc. On the whole, it includes 60 documents (~ 115000 tokens), where 30 documents are written in English and 30 – in Russian. In each subcorpora 25 documents were taken for the

training set, while the rest 5 formed the test set. The documents in each set were picked randomly. Obviously, in the UN documents no manually assigned keyphrases are provided, so we used document-headline pairs in the training set.

As it was already mentioned, KEA is a universal language-independent algorithm that means that the importance of a phrase for the document content does not depend on any particularities of a language. Although the realization of KEA allows to provide external language-dependent modules such as stemmers, for example. And its initial package contains stemmers for some languages, but Russian is not among them. As using different stemmers for document preprocessing could influence the resulting list of keyphrases, no linguistic processing of the documents was used in this experiment. Thus, equal conditions were set up for both languages.

In processing English texts we used an internal list of stopwords, created by the developers of the algorithm, and stopword list for the Russian language was collected from Russian National Corpus (RNC) lists of function words and abbreviations [8]. It includes the most frequent prepositions, particles, pronouns, interjections, some parentheses, digits and Latin characters.

For each document of the test set we obtained a list of 20 (the number recommended by the developers as containing the most salient keyphrases) the most relevant keyphrases. After that the lists were manually analyzed in order to find translation equivalents.

3.2 Results and evaluation

It is worth mentioning that results obtained in the course of experiments cannot be evaluated with high precision as the algorithms of keypharse extraction as such are hard to evaluate, especially when no manually assigned keyphrases are provided. Moreover, the algorithms like KEA, as a rule, work better for the documents that were preprocessed, – for languages with rich grammar like Russian in particular. As it was already noted, we did not perform preprocessing of the documents in our study to create at most equal conditions for both languages. Therefore, for each document we decided to calculate the percentage of semantically associated structural units for both outputs combined together. The number of units being members of some kind of semantic relations was dived by 40 (20 Russian keyphrases for a document and 20 English keyphrases for a document) and multiplied by 100 to get a percentage. Technically, of course, those are two different documents, but as our study is of semantic nature, we consider it to be unimportant detail. Obtained results with examples are shown in the Table 1.

Table 1. The percentage of semantically associated structural units of a document among Russian and English keyphrases.

Document id	The percentage of semantically associated structural units of a document	Examples

G1812398\400	65%	Paris Agreement – Парижского соглашения Annex I to the Convention – приложение I к Конвенции included in Annex – включенных в приложение
G1813678\80	67,5	TIR carnet holder – держателя книжки МДП subcontractor – субподрядчика container – контейнера
N1813436\38	65%	Advisory Committee – Консультативный комитет liquidation – ликвидации mission – миссии
N1813943\46	75%	terrorism and transnational organized crime – терроризмом и транснациональной организованной преступностью Security Council – Совет Безопасности crime – преступностью
V1802422\24	60%	member states – государство-член voting rights – права голоса contributions – взносов

As we can see, we indeed can find translation equivalents in the output what proves KEA's language-independence and new possibilities for research in that area.

Although for these figures some notes should be made. Firstly, KEA tends to break semantically associated units. For instance, for the document G1812398\400 we had *Paris*, *agreement* and *Paris Agreement* for both languages. It is quite a common issue for automatic keyphrase extraction, but among researchers there is still no convention how to conduct any kind of calculations in this case. In our paper we decided to count full phrases as well as their parts. So, in the example above, all three units were considered to be semantically associated.

Secondly, because of the certain nature of texts in our corpus, we mainly dealt with translation equivalents, and sometimes it is hard to tell, whether or not keyphrases are equivalent and whether the parts came from the same phrase. For example, for a document N1813943\46 were extracted *Совет Безопасности*, *Совет Безопасности напоминает*, *Security Council* and *encourages*. In such cases we had to turn to the original text, which is not very convenient within the experiment, because it was done manually for each document in the corpus, to look at the context. But it is still impossible to tell, if *Security Council* came from *Security Council encourages* or *Security Council recalls*. As a used corpus was not aligned, looking at the context becomes a separate problem.

Therefore, such, sometimes, high figures are a product of evaluation issues appearing while processing broken phrases. Those breaks may be caused not only by KEA's

peculiarities, but also by the absence of morphological preprocessing of the texts. It is commonly known that 'messy' data causes calculation mistakes, that is why we admit that our evaluation is raw and does not claim to be the only one possible or highly precise.

4 Automatic summarization of short stories

4.1 Data preprocessing

In this paper we used KEA to create extracts based on the original text [9, 10]. According to [11] extract is a collection of passages (ranging from single words to whole paragraphs) extracted from the input text(s) and produced verbatim as the summary.

For this experiment we compiled a corpus of 35 short stories written in Russian and Russian translations of famous literary works. Among the authors whose stories were used are A. Chekhov, O. Henry, D. Kharms and others. While selecting the only criterion was a small size. 30 short stories were used for the training set and the rest five for the test set. As manually assigned keyphrases for training we took abstracts for those stories written by users of [12].

Further actions can be divided in two ways:

1. Experiments based on the lemmatized training set:
   - lemmatization of the abstracts;
   - deleting stopwords;
   - lemmatization of the training set;
   - lemmatization of the test set;
   - extraction of 20 the most relevant keyphrases.
2. Experiments based on the non-lemmatized training set:
   - lemmatization of the test set;
   - extraction of 20 the most relevant keyphrases.
   - lemmatization of the output keyphrases.

The reason for this division is the fact that KEA produces different results depending on if the training set has been lemmatized or not. For lemmatization we used morphological analyzer pymorphy2 [13] in Python.

4.2 The algorithm

As the corpus has been processed and keyphrases for the test set extracted, an extract for a story is automatically composed based on obtained results. We developed and tested the algorithm which was implemented in Python. Our algorithm is composed of several modules including preprocessing as well as the module creating an extract.

The algorithm contains several stages:

1) the text is split into sentences: as the search of keyphrases in the text is conducted by lemmas, later we need to find and extract original sentences;
2) the title and the first sentence are extracted: we need the title to bound an extract with its story, and the first sentence gives it a start;

3) the search of the keyphrases in the sentences: at this point we have lemmatized original texts and their keyphrases to conduct a search by lemmas;
4) candidate sentences are assigned some scores (this stage will be discussed later);
5) selected sentences are extracted from the original text and the first five (including the first one) having a score more or equal to 2 form the extract.

Scores are assigned as follows:

1, if a keyphrase is included in one of the constructions listed below, and if it is a subject or a predicate of the sentence in the first two cases:

- noun:
  - noun + noun\verb\full adjective\short adjective (in the distance of +\- 1 from the main word)
- verb:
  - verb + noun\infinitive (in the distance of +\- 1 from the main word)
  - verb + full adjective + noun
- adjective:
  - adjective + noun (in the distance of + 1 from the main word)
  - verb + adjective + noun;

2, if a keyphrase in the sentence is among the first five from the output list;

3, if a sentence contains more than one keyphrase;

4, *5*, *6* are assigned for combinations and if a sentence contains several keyphrases.

4.3 Results and evaluation

Therefore, the obtained extracts are as follows.

Here is an extract for ‘Enemies’ by A. Chekhov. The story begins when to the doctor, whose son has just died, comes a visitor and asks for help because his wife is sick. The doctor refuses saying that he cannot work now, but eventually agrees to come.

Table 2. Examples of extracts automatically composed by the proposed algorithm.

Lemmatized training set	Non-lemmatized training set
ВРАГИ.	*ВРАГИ.*
В десятом часу темного сентябрьского вечера у земского доктора Кирилова скончался от дифтерита его единственный сын, шестилетний Андрей.	*В десятом часу темного сентябрьского вечера у земского доктора Кирилова скончался от дифтерита его единственный сын, шестилетний Андрей.*
	Кирилов, как был, без сюртука, в расстегнутой жилетке, не вытирая мокрого лица и рук, обожженных карболкой, пошел сам отворять дверь.
— Я дома, — ответил Кирилов.	*— Я дома, — ответил Кирилов.*

— Пока ехал к вам, исстрадался душой... Одевайтесь и едемте, ради бога... Произошло это таким образом.	*Очень рад, что застал... Бога ради, не откажите поехать сейчас со мной... У меня опасно заболела жена... И экипаж со мной... По голосу и движениям вошедшего заметно было, что он находился в сильно возбужденном состоянии.*
— Верьте, я сумею оценить ваше великодушие, — бормотал Абогин, подсаживая доктора в коляску.	*Когда Абогин еще раз упомянул про Папчинского и про отца своей жены и еще раз начал искать в потемках руку, доктор встряхнул головой и сказал, апатично растягивая каждое слово: — Извините, я не могу ехать... Минут пять назад у меня... умер сын... — Неужели?*
В его осанке, в плотно застегнутом сюртуке, в гриве и в лице чувствовалось что-то благородное, львиное; ходил он, держа прямо голову и выпятив вперед грудь, говорил приятным баритоном, и в манерах, с какими он снимал свое кашне или поправлял волосы на голове, сквозило тонкое, почти женское изящество.	

In this case, the second extract seems to be more appropriate, as it is more coherent and does not contain redundant information.

Now we can see a counter-example. The story is ‘Tobin’s Palm’ by O. Henry. Two friends are going to Coney Island to cut loose because one of them, Tobin, has just been deceived and robbed by his girlfriend. There they meet a gipsy who warns Tobin to stay away from certain people and says that he will meet a person who will bring him luck. So, the rest of the story Tobin and his friend are trying to find that person.

Table 2. Examples of extracts automatically composed by the proposed algorithm (continue).

Lemmatized training set	Non-lemmatized training set
Линии судьбы.	*Линии судьбы.*
Мы с Тобином как-то надумали прокатиться на Кони-Айленд.	*Мы с Тобином как-то надумали прокатиться на Кони-Айленд.*
Промеж нас завелось четыре доллара, ну а Тобину требовалось развлечься.	*Промеж нас завелось четыре доллара, ну а Тобину требовалось развлечься.*
Кэти Махорнер, его милая из Слайго,[70] как сквозь землю провалилась с того самого дня три месяца тому назад, когда укатила в Америку с двумя сотнями долларов собственных сбережений и еще с сотней, вырученной за продажу наследственных владений Тобина — отличного домишки в Бох Шоннаух и поросенка.	*Кэти Махорнер, его милая из Слайго,[70] как сквозь землю провалилась с того самого дня три месяца тому назад, когда укатила в Америку с двумя сотнями долларов собственных сбережений и еще с сотней, вырученной за продажу наследственных владений Тобина — отличного домишки в Бох Шоннаух и поросенка.*
— Я вижу дальше, — говорит гадалка, — что у тебя много забот и неприятностей от той, которую ты не можешь забыть.	*Ну и вот мы, я да Тобин, двинули на Кони — может, подумали мы, горки, колесо да еще запах жареных зерен кукурузы малость встряхнут его.*
— Берегись, — продолжает гадалка, — брюнета и блондинки, они втянут тебя в неприятности.	*Тобин выдает ей десять центов и сует свою руку, которая приходится прямой родней копыту ломовой коняги.*

Here, the first extract is likely to be more successfully made because it gives the story a start, while from the second one it is hard to understand what happened with characters after they had arrived at Coney Island.

To give estimation to obtained results, we asked 6 experts to evaluate the texts from the following three perspectives:

- which one of two extract variations is better: lemmatized or non-lemmatized; the one better is assigned 1 score, while the other gets 0 (further was evaluated the one that got 1 at this step);
- meaningfulness: if it is impossible to get something about a story from the extract, the score for this parameter equals 0; if a reader could get at least something, 1; and if an extract is for the most part clear, 2;
- preview: whether or not a given extract can be used as a preview for a short story.

The average evaluations for each parameter are shown in Table 3. As the first parameter is a matter of preference and refers to another issue (data preprocessing), total score was calculated only for 'Meaningfulness' and 'Preview' parameters, 3 consequently being the highest point..

Table 3. Expert evaluation of obtain results.

Title	lemmatized version	non-lemmatized version	Meaningfulness	Preview	Total score
Enemies, A. Chekhov	0,2	0,8	1,5	0,8	2,3
Strictly Business, O. Henry	1	0	1,5	0,8	2,3
Tobin's Palm, O. Henry	0,8	0,2	0,8	0,7	1,5
The Man in the Case, A. Chekhov	0,2	0,8	1,3	0,8	2,2
A story about a priest, M. Zoshchenko	0,2	0,8	1,2	0,8	2

Clearly, KEA can be used as an in-between tool for composing extracts for short stories, as it has shown competitive results, gaining the average total score more than or equal to 1,5 out of 3.

Interestingly, experts, as a rule, preferred a version based on non-lemmatized data. In a way it confirms our suggestion that stemming from the source package would be better for data preprocessing.

5 Conclusions

In this paper we tried to find and test some further applications of KEA, namely identifying translation equivalents in the same text written in several languages and summarizing short stories. As we can see, KEA has managed to find the equivalents in texts and summarize stories up to its preview. That means that KEA is capable to serve as a universal and effective tool for different tasks and may be useful not only for researchers but for naive users as well.

Acknowledgements

The reported study is supported by Russian Fund of Basic Research (RFBR) grants 16-06-00529 «Development of a linguistic toolkit for semantic analysis of Russian text corpora by statistical techniques».

References

1. Kaur, J., Gupta, V.: Effective approaches for extraction of keywords. In: International Journal of Computer Science Issues, vol. 7, № 6, pp. 144–148 (2010).
2. Beliga, S.: Keyword extraction a review of methods and approaches. URL: http://langnet.uniri.hr/papers/beliga/Beliga_KeywordExtraction_a_review_of_methods_and_approaches.pdf (2014).
3. Sokolova, E. V., Mitrofanova, O. A.: Automatic Keyphrase Extraction by applying KEA to Russian texts. In: IMS 2017 Proceedings, St.-Petersburg (2017).
4. Nenkova, A., McKeown, K.: Automatic summarization. In: Foundations and Trends in Information Retrieval, vol. 5, № 2–3, pp. 103–233. (2011).
5. KEA Homepage, http://www.nzdl.org/Kea/index.html, last accessed 2018/05/27.
6. Witten, I.H., Paynter G.W., Frank E., Gutwin C., Nevill-Manning C.G.: KEA: Practical Automated Keyphrase Extraction. In: Design and Usability of Digital Libraries: Case Studies in the Asia Pacific, IGI Global, pp. 129– 152 (2005).
7. The United Nations Homepage, http://www.un.org/ru/index.html, last accessed 2018/05/27.
8. RNC Homepage, http://www.ruscorpora.ru/, last accessed 2018/05/27.
9. Kazantseva, A., Szpakowicz, S.: Summarizing short stories. In: Computational Linguistics, vol. 36, № 1, pp. 71–109 (2010).
10. Luhn, H.P.: The automatic creation of literature abstracts. In: IBM Journal of research and development, vol. 2, № 2, pp. 159–165 (1958).
11. Hovy, E., Lin, C.Y.: Automated text summarization and the SUMMARIST system. In: Proceedings of a workshop on held at Baltimore, Maryland: October 13–15, 1998, Association for Computational Linguistics, pp. 197–214 (1998).
12. FantLab Homepage, https://fantlab.ru/, last accessed 2018/05/27.
13. Korobov, M.: Morphological Analyzer and Generator for Russian and Ukrainian Languages. In: Analysis of Images, Social Networks and Texts, pp 320–332 (2015).