Czech Keyword Extraction in Perl

Tadeas Jun

November 2021

1 Introduction

Keyword extraction is a relatively well-studied field, generally using several main algorithms either using or not using machine learning. Machine learning methods generally use large database of training data which the AI uses to effectively learn what a given type of text could consider as keywords.

When working on this project, I did not have access to any training data. Furthermore, it was not specified what document type the code would analyze. Because of this, I had to opt to alternative methods.

One of the widely-used keyword extraction algorithms is tf-idf (term frequency-inverse document frequency), which extracts keywords from one document out of a collection of documents. The algorithm compares the word frequency of a given word in the selected document with its frequency in all of the other documents. It later uses this data to calculate the relative importance of the given word.

In my solution, I used an edited version of the tf-idf algorithm, which compares the word frequency to the entire Czech Corpus, instead of other documents in a given database.

This paper shortly expands on the theory behind this single-document keyword extraction algorithm, as well as the practical solution in the Perl language. It also showcases some of the results and mentions some ways the code could be further improved.

This documentation is a rough translation of the original Czech version, available at */documentation/cze/dokumentace.pdf*. It is notable to mention that the algorithm and its code are tailored to work for the Czech language.

2 Theory

This section details the theory of the algorithm, going through the text preparation, working with the corpus, and assigning a given importance value to every relevant word.

2.1 Preparing the text

In theory, the edited tf-idf algorithm should work without any significant edits to the input text. However, for the purposes of this exercise, preparing the text and removing words that have a low probability of being of any importance heavily lowers the time required to finish the process. The text preparation transforms a large list of words into a fraction of the most-probably important ones. In Example 1 in the Results section, the text preparation phase reduced the word list from the original 11 524 words to a total of 345, which the algorithm could then individually rate.

Before starting the preparation process, the program needs to load the Czech corpus and the input document itself. The corpus is taken from the Czech National Corpus (Český Národní Korpus) [1] - it is "a representative corpus of contemporary written Czech" [2] including over 100 000 000 tokens (individual recorded words) and data about their frequencies. The input document itself is then loaded from the defined input, split into individual words, and special characters (periods, commas, numbers, parentheses...) are removed from it.

The first step of the text preparation phase is the removal of stop words. These are words that appear the most often in the Czech language; words such as "a" ("and"), "ale" ("but"), "na" ("on"), "jsem" ("I am"), etc. Words like these usually do not hold any meaning by themselves, so they can be removed from the list. In this specific solution, the code cuts off the 150 most common Czech words - testing revealed that with a higher threshold, the program sometimes started removing important words from the input text.

Next, words with 3 or less characters do not connote any meaning in the vast majority of cases in Czech. These words are therefore removed as well - they are usually abbreviations or stop words left over from the last step.

Before moving on to the last step of the preparation process, the algorithm counts the frequency of each leftover word in the input text. Finally, it removes words that do not appear in the text very often, since those words have a very low probability of being keywords. The threshold of frequency

is calculated dynamically using the formula

$$\frac{log_e(n)}{log_e(10)} + 0.5$$

where n is the total number of analyzed words. The result of this calculation is rounded to the nearest integer, which is then used as the frequency threshold; all words with less occurrences are removed from the list. Using this formula means that the longer the text, the more common must words be to stay in the list. However, since the scale is logarithmic, the threshold does not rise too fast with longer documents.

If a word is not included in the corpus at all, it is also removed.

After these steps, the list of words is reduced enough to be prepared to be individually rated based on assumed importance.

2.2 Assigning importance values

Once the text is prepared and the word list is reduced to its limit, the algorithm can start assigning importance values to each individual word - this value represents the estimated importance of a given word relative to the input document as a whole. The algorithm loops over the list, calculating the importance value for each word using the given formula

$$\frac{x_i}{n_a} \cdot log_e(\frac{\frac{n_d+n_c}{2}}{1+c_i} + 1)$$

where n_a is the amount of analyzed words in the document, x_i is the frequency of the given word in the document, n_d is the sum of frequencies of all analyzed words, n_c is the total number of unique words in the corpus (334 833 in this case), and c_i is the frequency of the given word in the corpus.

The values are only relative, so, in the following step, they are normalized to the range of 0.5 - 100 and rounded to two decimal places, to increase human readability.

The word list is then sorted by the importance values in descending order. The code prints the first 20 words (or all the words, if there is less than 20 of them) into the standard or defined output. If the user did not select the simplePrint option (see User documentation), it prints out the importance values of the individual words as well.

3 Code

The klicova_slova.pl program is written in the Perl language, version 5.30.0 built for x86_64-linux-gnu-thread-multi. It uses the CPAN libraries Text::CSV_XS::TSV to load the corpus, List::Util for the text preparation phase, and Getopt::Long to load the input parameters. It includes 8 functions that are gradually called for the preparation, analysis, and printing out phases. Following is a short description of all the functions.

The GetAllWords function loads the input text from the defined input, line-by-line, and splits them into individual words (characters separated by a space or another whitespace character). It then changes the word to lowercase, and removes all special characters from it, using the following list:

Figure 1: The list of special characters to be removed.

The individual words are then saved into the @wordList array.

GetCzechCorpus is a function that loads the full Czech corpus from the syn2015_word_utf8.tsv file. It saves the corpus into an array of hashes with the elements rank, word, and frequency.

After the input text and the corpus are loaded, the algorithm starts the text preparation phase, first using the RemoveStopWord function. The function removes the 150 most common Czech words from the @wordList, using two nested grep statements, filtering the @wordList array via a secondary array, @frequentCorpusWords.

The next function, RemoveShortWords, uses the grep function to remove of words in which the length (in characters) is less or equal to 3.

The GetWordFrequencies function counts the frequency of each of the remaining words in the edited input text and saves them into the %frequencies hash. The individual words, serving as the hash keys, are later on sorted by the frequency and saved into the @sortedKeyFrequencies array.

The RemoveUnusualWords function calculates the cut-off threshold for low frequency words and removes those words from the list. The formula used to calculate the threshold is showcased and explained in the Theory section.

The CalculateImportanceValues function uses the aforementioned formula to come up with one value to represent the importance of the word relative to

the text. This function is the most time consuming - for longer texts (e.g. novels) it can take over a minute. After all the values are calculated, the words are sorted by their importance.

The last function is NormalizeImportances, which normalizes the values to the 0.5 - 100 range and rounds them to 2 decimal places.

After the code completes all these functions, it prints the results out using a simple for loop.

4 Results

To showcase some of the results achieved when testing the algorithm, the paper lists the results of 3 examples - the full text of the short novel The Little Prince (Malý Princ) by Antoine de Saint-Exupéry [3], and the contents of two Czech Wikipedia articles on Fighter aircrafts (Stíhací letoun) [4] and Heredity (Dědičnost) [5]. Following is a summary of the results.

4.1 The Little Prince (Malý princ)

Total word count in the document: 11 524

Total removed stop words: 3 691

Total removed short words: 1 214

Total relevant unique words: 3 074

Total removed infrequent unique words: 2 729

Total analyzed unique words: 345

Result keywords, in descending order by importance (Czech):

princ malý prince beránka planetu malého král baobaby planeta trny beránek sopky pijan princi bydlil slunce svítilnu dodal trochu zasmál

Result keywords (translated):

prince little lamb planet king baobabs thorns volcanoes drunkard lived sun lantern added bit laughed

4.2 Fighter aircraft (Stíhací letoun)

Total word count in the document: 2 316

Total removed stop words: 597

Total removed short words: 150

Total relevant unique words: 1031

Total removed infrequent unique words: 935

Total analyzed unique words: 96

Result keywords, in descending order by importance (Czech):

stíhací stíhačky letouny záchytné letoun války bombardéry lightning stroje stíhaček fighter např světové suchoj stíhače přepadové generace lockheed výzbrojí studené

Result keywords (translated):

fighter jets aircrafts interceptor wars bombers lightning machines e.g. world suchoj generations lockheed arms cold

4.3 Heredity (Dědičnost)

Total word count in the document: 735

Total removed stop words: 177

Total removed short words: 31

Total relevant unique words: 392

Total removed infrequent unique words: 366

Total analyzed unique words: 26

Result keywords, in descending order by importance (Czech):

dědičnost dědičnosti sekvence recesivní heterozygotní dominantní geny rozmnožování alely buňky generace dědičné přenos křížení organismů dělení organismu nazývá informace rozdíly

Result keywords (translated):

heredity sequence recessive heterozygote dominant genes procreation alleles cells generations passing breeding organism division called information differences

5 Further improvements

In the Results section, you might already notice a few shortcomings of the code. The abbreviation "např" ("for example") appears in highlighted keywords of academic articles often - it would be possible to remove all Czech abbreviations in the text preparation phase by removing all words that end with a period, after which does not follow a capital letter.

Furthermore, one of the large decisions made during the work on this project was the question of lemmatization. Lemmatization is the process of grouping together the inflected forms. For example, in the Little Prince example, the extracted keywords contain the words "princ", "prince", and "princi". In Czech, all of these words refer to the same subject - therefore it would be possible to group them under the common lemma, "princ". However, the initial purpose of this algorithm was to be used as a tool to search for similarities in documents; lemmatization of the keywords would require the target database of documents to also be lemmatized, thus adding a complicated step to the similarity-search process.

In any case, if lemmatization was to be implemented, the code would use the lemmatized corpus syn2015_lemma_utf8. For the lemmatization itself, for example, the CPAN library Ufal:MorphoDiTa could be used.

6 Conclusion

The program successfully extracts keywords from Czech documents. Generally, it lends the best results in documents of around 2 000 words. The code was tested on both fiction and non-fiction (in the form of Wikipedia articles).

While the code could be improved in several ways, as outlined in the Further improvements section, the Results section shows the code extracts keywords that are relevant to the input text fairly reliably.

References

[1] M. Křen, V. Cvrček, T. Čapka, A. Čermáková, M. Hnátková, D. Kováříková, H. Skoumalová, P. Truneček, P. Vondřička, and A. Zasina, *SYN2015: reprezentativní korpus psané češtiny*. Praha: Ústav Českého národního korpusu FF UK, 2015.

[2] V. Cvrček, "Korpus SYN2015", *Wiki Český Národní Korpus*. [Online]. Available: https://wiki.korpus.cz/doku.php/cnk:syn2015.

[3] A. de Saint-Exupéry and R. Podaný, *Malý princ*, 14. ed. Praha: Albatros, 1943.

[4] "Stíhací letoun", *Wikipedia: the free encyclopedia*, 2007. [Online]. Available: https://cs.wikipedia.org/wiki/St%C3%ADhac%C3%AD_letoun.

[5] "Dědičnost", *Wikipedia: the free encyclopedia*, 2017. [Online]. Available: https://cs.wikipedia.org/wiki/D%C4%9Bdi%C4%8Dnost.