

USING THE PYTHON LIBRARY WHEN CLASSIFYING SCIENTIFIC TEXTS

Ismukanova A.N.¹, Lavrov D.N.² (Russian Federation), Keldybekova L.M.³,
Mukumova M.Zh.⁴ (Republic of Kazakhstan) Email: Ismukanova346@scientifictext.ru

¹Ismukanova Aygerim Nauryzbayevna - Postgraduate Student;

²Lavrov Dmitry Nikolaevich - PhD in Engineering, Associate Professor, Head of the Department,
DEPARTMENT OF COMPUTER TECHNOLOGIES AND NETWORKS, FACULTY OF COMPUTER SCIENCES,
OMSK STATE UNIVERSITY NAMED AFTER F.M. DOSTOYEVSKY,
OMSK;

³Keldybekova Liliya Muratbekovna – Teacher;

⁴Mukumova Marzhan Zhumanovna – Teacher,
DEPARTMENT OF FOREIGN LANGUAGES, FACULTY OF PHILOLOGY,
KOKSHETAU STATE UNIVERSITY NAMED AFTER SH. UALIKHANOV,
KOKSHETAU, REPUBLIC OF KAZAKHSTAN

Abstract: the classification of scientific texts in the Russian and Kazakh languages, by means of assignment of a universal decimal code (UDC) is an actual problem nowadays. The problem of the classification of scientific texts is easily solved for the English language due to its simplicity of morphology and syntax. On this way there is a number of unresolved tasks for the Russian language and practically the usage of analogical reception for the Kazakh language is not investigated. For the Russian language several researches of applicability of different approaches were conducted [1-4].

New technologies for the LSA model could represent a important advance of the assessment of scientific texts. LSA model despite the complexity of the opacity and can be used for a number of different tasks with a generalization or extension of the meaning of the search query.

Keywords: latent semantic analysis (LSA), machine learning (ML), classification, NLTK library.

ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ PYTHON ПРИ КЛАССИФИКАЦИИ НАУЧНЫХ ТЕКСТОВ

Исмуканова А.Н.¹, Лавров Д.Н.² (Российская Федерация), Кельдыбекова Л.М.³,
Мукумова М.Ж.⁴ (Республика Казахстан)

¹Исмуканова Айгерим Наурызбаевна – аспирант;

²Лавров Дмитрий Николаевич - кандидат технических наук, доцент, заведующий кафедрой,
кафедра компьютерных технологий и сетей, факультет компьютерных наук,
Омский государственный университет им. Ф.М. Достоевского,
г. Омск;

³Кельдыбекова Лилия Муратбековна – преподаватель;

⁴Мукумова Маржан Жумановна – преподаватель,
кафедра иностранных языков, факультет филологии,
Кокшетауский государственный университет им. Ш. Уалиханова,
г. Кокшетау, Республика Казахстан

Аннотация: классификация научных текстов на русском и казахском языках посредством присвоения им универсального десятичного кода (УДК) является актуальной задачей. Задача классификации научных текстов прекрасно решается для английского языка в силу простоты морфологии и синтаксиса этого языка. На данном пути имеется ряд нерешенных задач для русского языка и практически не исследовано использование аналогичных приемов для казахского языка. Для русского языка проводились несколько исследований применимости разных подходов.

Новые технологии, для модели LSA (латентного-семантического анализа), могли представлять важное усовершенствование в исследовании оценки научных текстов.

Модель LSA, несмотря на трудоемкость и непрозрачность, может использоваться для разного ряда задач при обобщении или расширении смысла поискового запроса.

Ключевые слова: латентный семантический анализ (ЛСА), машинное обучение (МА), классификация, библиотека NLTK.

UDC 004.89

Python is a dynamic programming language. It interprets object-oriented text model. On Python in combination with library for works with natural languages NaturalLanguageToolkit (NLTK) the important component in systems on machine learning is implemented. Python uses standard libraries (for example: NumPy and SciPy) for scientific research, mathematical calculations and also engineering decisions. Standard manager

of packages in Python is Python Package Index (pip). This program is used for addition of libraries in a program system.

“Pininstallnltk” is a command for adding NLTK library in our program. Command "Pip freeze" is used to display the list of libraries in the program.

In Python command “virtualenv” is used for creation of the separate environment, the instrument of execution of programs, including certain instructions Python and sets of libraries. Creating virtual environment for execution of the program, we add the following libraries: feed parser, numpy, scipy, NLTK [1].

Categories in Python and NLTK

In the beginning it is necessary to process imitating news feeds RSS, to analyze scientific information with the help of Naive Bayes Classifier and to clarify them on category by means of an algorithm kNN. Then we turn on NLTK.

NLTK is an excellent library for processing texts in natural languages, the ready - loaded sets of basic data in the form of the “body” and also program interfaces for convenient access to these data are applied to it. To install the “body”, you need to execute the command provided below, more than 10 000 news messages will be loaded into your catalog ~/nltk_data/corpora/. Categorized data are suitable for imitation of a news feed.

Listing 1. Import of NLTK

```
$python .....# we enter an interactive cover
>>> importnltk .....# we import library NLTK
>>>nltk.download().....# we start the loader and we enter "d" Identifier>reuters.....#.we specify the
"body".
```

The file nltk_/data/corpora/reuters/cost.txt. contains the list of names of files with notes and also the categories appointed to each of files.

There is a problem of often-used , but actually non-significant words such as articles, conjunctions and prepositions. These auxiliary words complicate work. So, a natural language is rather various and it needs cleaning before work. Python and NLTK allow by means of a method "normalized-words" from RssItem to exclude all these non-significant words. In particular NLTK cleans the crude text of articles from the built-in tags by means of only one line of a code. Besides, by means of regular expression it is carried out removals of a punctuation, then the text is divided into words and is transferred to a lower "body".

Listing 2. Class RssItem

```
class RssItem:
    ...regex = re.compile('[%s]' % re.escape(string.punctuation))
    ...defnormalized_words(self, article_text):
        words = []
        oneline = article_text.replace('\n', ' ')
        cleaned = nltk.clean_html(oneline.strip())
        toks1 = cleaned.split()
        for t1 in toks1:
            translated = self.regex.sub("", t1)
            toks2 = translated.split()
            for t2 in toks2:
                t2s = t2.strip().lower()
                if self.stop_words.has_key(t2s):
                    pass
                else:
                    words.append(t2s)
        returnwords
```

The list of auxiliary words are from NLTK in one line and also other natural languages are supported. NLTK.corpus.stopwords.words("kazakh ").

NLTK provides several classes of the morphological analysis for further normalization of words.

Classification by a Simple Bayesian Algorithm

The algorithm NaiveBayes, a simple Bayesian algorithm, is widely-known and built-in in NLTK in the form of a class NLTK.NaiveBayesClassifier. Bayes allows to classify elements upon existence or lack of certain elements in their structure. As elements, certain words of a natural language are used. The algorithm is not difficult, it does not mean interrelations between elements. There is a certain class nltk.probability.FreqDist in NLTK. It helps to define often found words. The following method 'collect-all-word' return the massif containing all words from all training notes. The massif is processed by method 'identify_rop_words' and returns often found words. Keys of function of a class of nltk.FreqDist actually are sorted according to the corresponding values, thereby it is possible to select the 1000 most often found words, specifying the range of indexes according to syntax Python [3].

Listing 3. The Usage of a class nltk FreqDist

```

defcollect_all_words(self, items):
    all_words = []
    for item in items:
        for w in item.all_words:
            words.append(w)
    return all_words
defidentify_top_words(self, all_words):
    freq_dist = nltk.FreqDist(w.lower() for w in all_words)
    returnfreq_dist.keys()[:1000]

```

For imitation of RSS on the NLTK database from scientific texts of Russian and Kazakh it is necessary to allocate and read the file, for this purpose: / nltk_data/ corpora/ rurers/cats.txt/:

```

defread_reuters_metadata(self, cats_file):
    f = open(cats_file, 'r')
    lines = f.readlines()
    f.close()
    returnlines

```

Obtaining the characteristics of each message from a tape is carried out by method ~features~ from a class RssItem. Using this method the massif at first of all words (all_words) of texts is reduced to a set of unique words, smaller by the value , due to elimination of similar words. To check in scientific texts of existence or lack of widespread words it is necessary to execute pass according to the list.

Listing 4. The corresponding Code on Python

```

deffeatures(self, top_words):
    word_set = set(self.all_words)
    for w in top_words:
        features["w_%s" % w] = (w in word_set)
    returnfeatures

```

For processing of an algorithm it is necessary to create a training set of messages and to collect their individual characteristics. Training of classification holds exactly one place of a code.

Listing 5. Training nltk.NaiveBayes classifier

```

defclassify_reuters(self):
    ...
    training_set = []
    for item in rss_items:
        features = item.features(top_words)
        tup = (features, item.category) # tup is a 2-element tuple
        featuresets.append(tup)
    classifier = nltk.NaiveBayesClassifier.train(training_set)

```

One of the methods to distinguish interrelations between words is to join in a set of parameters widespread phrases from two (bigrams) and three words (trigrams). In NLTK there is a support of these opportunities in the form of functions nltk.bigrams(...) and nltk.trigrams(...). Just at the library chose from all data set N of the most often found words, it can identify the most popular two and three-word phrases and use them as parameters.

The Algorithm k-NearestNeighbours

The most demanded algorithm for creation of the recommendations is the algorithm k-NearestNeighbour. This algorithm creates the list of categories and compares to each category a set of information and data. Then defining coinciding elements , algorithm compares a set of data. A set of information is provided by a set of numerical values. The algorithm kNN issues the whole set of guidelines with degrees, on the basis of sets of scientific texts. In the work as processing of tapes of RSS, the values of tags coincided with categories, and processing of scientific texts represents work with massive values for 1000 often found words. Creating of each word in the massif of value can represent the Boolean value (0 pr 1), frequency of occurrence of words as a percentage or in figures, exponential expression from frequency or will be defined by other value. For tools of the developer the Python programming language is a laconic and powerful language. It well realizes machine learning, processes a natural language [2].

Tokenization Function in Python. Tokenization is called a division of data into small parts , i.e. tokens. Words and punctuation signs belong to tokens. Texts are represented in the form of massif of necessary words. Then in massif of significant words we make cleaning regarding signs of a punctuation and non-significant words.

Listing 6. The Functions of Tokenization

```

# -*- coding: utf-8 -*-
import nltk
import string

```

```

from nltk.corpus import stopwords
def tokenize_me(file_text):
    #firstly let's apply nltk tokenization
    tokens = nltk.word_tokenize(file_text)
    #let's delete punctuation symbols
    tokens = [i for i in tokens if ( i not in string.punctuation )]
    #deleting stop_words
    stop_words = stopwords.words( 'kazakh' )
    stop_words.extend(['не', 'мынау', 'солай', 'онда', 'болу', 'қалай', 'мен', '—', 'оған', 'ал'])
    tokens = [i for i in tokens if ( i not in stop_words )]

    #cleaning words
    tokens = [i.replace("«", "").replace("»", "") for i in tokens]
return tokens

```

A standard function from Python library - nltk word_tokenize() was used for work many languages, but we needed only the Russian and Kazakh languages. The following step we destroy punctuation mark, checking each word string.punctuation. Next a set of auxiliary words in the Russian and Kazakh languages we import to word stop from library for work with nature languages NLTK. Having removed from the scientific text all significant words and punctuation marks in a final stage, we receive the list of words - the necessary terms for further processing. On machine learning similar systems are implemented on Python with library for works with scientific texts NaturalLanguageToolkit (NLTK) and their classifications [1-4].

References / Список литературы

1. *Moraes R., Valiati J.F., Gavião Neto W.P.* Document-level sentiment classification: An empirical comparison between SVM and ANN. Expert Systems with Applications. 2013, № 40. Pp. 621–633.
2. *Pontiki M., Galanis D., Pavlopoulos J., Papageorgiou H., Androutsopoulos I., Manandhar S.* SemEval-2014 Task 4: Aspect based sentiment analysis. Proc. 8th Int. Workshop on Semantic Evaluation (SemEval 2014). Dublin, Ireland, 2014, Pp. 27–35.
3. *Medhat W., Hassan A., Korashy H.* Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journ., 2014. № 5. Pp. 1093–1113.
4. *Polyakov I.V., Sokolova T.V., Chepovsky A.A., Chepovsky A.M.* Text classification problem and features set. Vestn. NGU. Ser.: Informatsionnye tekhnologii [Novosibirsk State Univ. Journ. of Information Technologies], 2015. Vol. 13. Iss. 2, Pp. 55–63 (in Russ.).