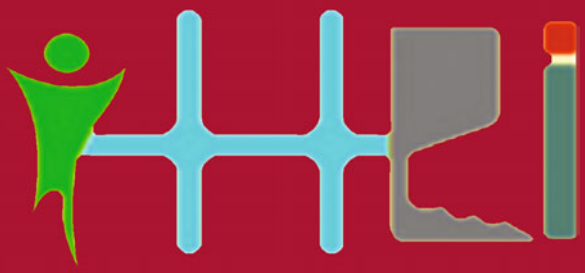
Hakimjon Zaynidinov
Madhusudan Singh
Uma Shanker Tiwary
Dhananjay Singh (Eds.)

LNCS 13741

Intelligent Human Computer Interaction

14th International Conference, IHCI 2022
Tashkent, Uzbekistan, October 20–22, 2022
Revised Selected Papers



Springer

Lecture Notes in Computer Science **13741**

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.

LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.

Hakimjon Zaynidinov · Madhusudan Singh ·
Uma Shanker Tiwary · Dhananjay Singh
Editors

Intelligent Human Computer Interaction

14th International Conference, IHCI 2022
Tashkent, Uzbekistan, October 20–22, 2022
Revised Selected Papers

Springer

Editors
Hakimjon Zaynidinov
Tashkent University Information Technologies
Tashkent, Uzbekistan

Uma Shanker Tiwary
Indian Institute of Information Technology
Allahabad, India

Madhusudan Singh
Oregon Institute of Technology
Klamath Falls, USA

Dhananjay Singh
Hankuk University of Foreign Studies
Yongin, Korea (Republic of)

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-031-27198-4 ISBN 978-3-031-27199-1 (eBook)
https://doi.org/10.1007/978-3-031-27199-1

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Methods for Creating a Morphological Analyzer

Elov Botir Boltayevich(✉), Hamroyeva Shahlo Mirdjonovna, and Axmedova Xolisxon Ilxomovna

Tashkent State University named after Alisher Navoi University of Uzbek Language and Literature, Tashkent O'qituvchi Street 103, Tashkent, Uzbekistan
{elov,a.xolisa}@navoiy-uni.uz

Abstract. The morphological analysis process is an important component of natural language processing systems such as spelling correction tools, parsers, machine translation systems, and electronic dictionaries. This article describes the stages of a text analyzer, methods for creating a morphological analyzer and a morphological generator. Ways to use the NLTK package tools in Python when creating a morphological analyzer, examples of software codes are given. Also, morphological analyzer structure and architecture are presented on the basis of the morphological analysis process (flexion, derivative, affixpids detection, compound forms).

Keywords: Natural language processing · NLP · Python · Morphological analyzer · Token · Lemmatization · Stemming · Morphological generator · Search engine · Stemming algorithm · PorterStemmer

1 Introduction

Morphology is a section that studies the grammatic meanings of words through morpheme. Morpheme is the smallest unit of language that can not be divided into meaningful parts. This article analyzes the issue of creating morphological analyzer and morphological generator for languages other than English using stemming and lemmatization, stemmer and lemmatizer, machine learning tools, search engines. Today, a number of scientists around the world are conducting scientific research on the creation of a morphological analyzer. In particular, Adnan Öztürel, Tolga Kayadelen and ISIN Demirsahin presented a broad coverage model of Turkish language morphology and an open source morphological analyzer that implements it [1]. This model covers the subtle aspects of Turkish morphology, syntax, from which it can be used as a guide in the development of a language model. The Model performs Turkish morphotactic using OpenFst as finite state transducers and Morphophonemic processes Thrax grammatics. Arabic linguists Y. Jeefer and K. Bouzoubaa Arabic Morphological Analyzers presented the methods of use in syntactic analysis programs, search engines and machine translation systems.

2 Materials and Methods

Natural Language understanding the first version of the Arabic language corps dedicated to the evaluation of Natural Language was created [2]. The four most common and

H. Zaynidinov et al. (Eds.): IHCI 2022, LNCS 13741, pp. 27–38, 2023.
https://doi.org/10.1007/978-3-031-27199-1_4

most advanced morphological analyzers (Hunmorf-Okamorf, Hunmorf-Foma, Humor and Hunspell) for the Hungarian language were analyzed, compared by G. Szabó, L. Kovács. They compared the token systems of annotation instead of the lemmatization properties of analyzers [3].

1. NLPni ikkita asosiy komponentga ajratish Understanding, NLU).
2. Tabiiy tini generatsiya qilish (Natural Language Generation, NLG).

The steps required in the performance of NLP tasks will be considered. The source of the natural language can be speech (sound) or text. Stages of text processing are presented in Fig. 1.

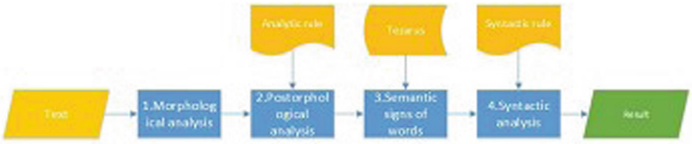


Fig. 1. Stages of text analyzer

In the field of Uzbek computer linguistics, morphological analyzer and morphological generator has not yet been created. In this article, we will consider ways to solve this issue based on world experience.

When creating a morphological Analyzer, It is worthwhile to study the following concepts:

1. morphological units;
2. stemmer;
3. lemmatization;
4. development of Stemmer in Uzbek language;
5. Morphological Analyzer;
6. Morphological Generator;
7. search engine development.

3 Morphological Units

Morpheme is the smallest meaningful part of the language. Morpheme is divided into two types: the limbs are also referred to as free morphemes, since they can also be used without adding affix (suffix) in sentences. (Additional) are not used separately in the language. Because they can not have a lexical meaning in an independent forms and always exists together with independent morphemes. Let's look at the word unbelievable in English. Here the believe is a predicate or a free morpheme. It means quot; independently; flour and able morphemes are affiks (addition) or paired morphemes. These morphemes can not exist in an independent form, they do not mean, they are used together with the core [22, 23].

Natural languages according to the formation of the grammatic form are divided into the following groups:

1. (isolating languages) – Languages of the peoples of South East Asian countries [24];
2. (agglutinative languages) – Turkic, bantu, Mongolian, Finnish-Ugric languages [22, 25];
3. (inflecting languages) – Indo-European and families of som languages [26, 27].

1. The dictionary composition of these languages consists mainly of singlestringed limbs, which do not have features of punctuation, punctuation. Therefore, in these languages word-building exercises, there are only loads that perform the function of auxiliary words. In amorphous languages, words consist only of independent morphemes; they do not include the data of the time (past, present and future) and the number (Unit or plural).
2. In agglyutinative languages, words will consist of a suffix and a suffix attached to it; the morphological composition of the word (suffix) is clearly distinguished. Bunda represents each additional separate meaning, task. For example, in Turkic languages, including Uzbek, the forms of words and words are formed by the addition of suffixes to the basis with a certain consistency.
3. Flektiv languages – it is characterized by the fact that the appendages merge with the core and are absorbed into it. In such languages, thematikmatic meanings are expressed by inflection, book in Arabic (unit) (plural). In Russian, the drug (unit) is a druzya (plural). Flektiv languages include IndoEuropean and som language families. Flektiv languages are divided into synthetic and analytical languages. In synthetic languages, the grammatic meanings (interaction of words in the sentence) are expressed by means of form-forming suffixes (eg., Russian, German). In analytical languages, the grammatic meanings are expressed not by means of word forms (form- forming suffixes), but by means of auxiliary words, word order, tone(English, French, Spanish). In Flektiv languages, words are divided into simple units, all these simple units show different meanings.
4. Polisynthetic languages– the main unit of speech is the word-sentence. There can not be a strict limit between the classified languages, since some language phenomena that occur in one language can also occur in others. For example, Oceanic languages can be described as both amorphous and agglyutinative languages [28].

Morphological (typological) classification of languages it is important to group the languages of the world according to certain morphological features, to create a general drawing of them [29, 30, 31].

Morphological processes are divided into the following others [32]:

1. (inflection);
2. (derivation);
3. (semiaffixes);;
4. (combining forms);
5. (cliticization).

On the basis of this stage, the structure of the work of the morphological analyzer of the Uzbek language is shown in Fig. 2 below:

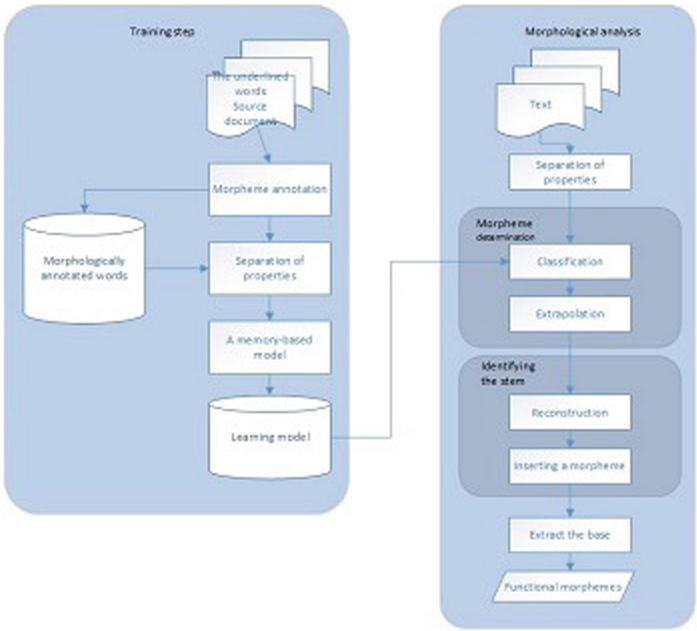


Fig. 2. Morphological analyzer structure

At the stage of infection, the word is transformed into a form that expresses the person, Number, time, gender and other grammatic categories. At this stage, the syntactic category of the lexeme remains unchanged.

At the stage of derivation, the syntactic category of the word changes. At the stage of determining semifixes, morphemes are formed in the form of compound words, abbreviations. For example: noteworthy, antisocial, anticlockwise, etc.

Stemmer: In the process of stemmatization, the determination of the word core is carried out by removing suffixes from the word [33–35]. For example, the question from nature is analyzed as follows: from [nature] (conjugate suffixes Yas forming a syntactic form)

Search engines (Google, Yahoo, Yandex) use stemming mainly in identifying cores and storing them as indexed words in order to increase the accuracy of the search for information. Search engines call words that have the same meaning as synonyms, and this query can be some kind of more accurate implementation of the results.

Currently, many moroanalysts are using the stemming algorithm, developed by Martin Porter. This algorithm is designed to replace and analyze suffixes that are present mainly in English words. To implement stemming in the NLTK package, we can create a copy of the PorterStemmer class, and then execute stemming by calling the stem() method

(Fig. 3). In NLTK, we will consider the stemming process using the PorterStemmer class [36, 37]:

```
    import nltk
from nltk.stem
import PorterStemmer
stemmerporter = PorterStemmer()
print(stemmerporter.stem('removing'))
print(stemmerporter.stem('happies'))
result
    remov
    happi
```

PorterStemmer the class has a lot of vocabulary and word forms in English. The process of determining the cores consists of several stages: the word shorter becomes a word or a form with a similar meaning. The Stemmer I interface defines the stem () method and all Stemmers are sampled from the Stemmer I interface [36].

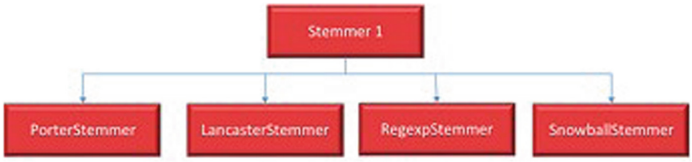


Fig. 3. Stemming methods in the NLTK package

Another stemming algorithm, known as the Lancaster stemming algorithm, was developed at Lancaster University. Similar to the PorterStemmer class, the Lancaster Stemmer class is also used to implement Lancaster stemming in NLTK. But the main difference between the two algorithms is that Lancaster stemming involves the use of more words of different characteristics in comparison with Porter Stemming.

```
    from nltk.stem import LancasterStemmer
stemmerlan=LancasterStemmer()
print(stemmerlan.stem('remov' ))
print(stemmerlan.stem( 'happies'))
```

```
remov
happy
```

We can also build our own Stemmer in NLTK using Regexpstemmer. It works by taking a line and removing the prefix or suffix of the word when compatibility is found. Let's look at an example of stemming using Regexpstemmer in NLTK:

```
    from nltk.stem import RegexpStemmer
stemmerregexp=RegexpStemmer('ing')
print(stemmerregexp.stem('removing' ))
print(stemmerregexp.stem( 'happiness'))
print(stemmerregexp.stem( 'pairing'))
```

```
remov
happiness
pair
```

We can use Regexpstemmer in cases where it is not possible to detect the core using PorterStemmer and LancasterStemmer. SnowballStemmer is used to perform stemming in 13 languages apart from English. To perform stemming with the help of snowststemmer, first of all, it is necessary to specify tilni where the stemming should be performed. Then, using the steam() method, stemming is performed. With the help of snowststemmer, the process of stemming in NLTK in Spanish and French is carried out as follows:

```
    import nltk
from nltk.stem import SnowballStemmer
print(" ".join(SnowballStemmer.languages)) //see which languages are supported
spanishstemmer=SnowballStemmer('spanish') // language selection
print(spanishstemmer.stem( 'comiendo'))
frenchstemmer=SnowballStemmer( 'french') //language selection
print(frenchstemmer.stem('manger')) // word steam
```

Let's look at the following code available, which allows us to implement stemming:

```
    Class StemmerI(object):
"""
"""
def stem(self, token):
"""
"""
raise NotImplementedError()
```

With the help of several Stemmers, we will consider the code that will be used to perform stemming:

```
import nltk
from nltk.stem.porter import PorterStemmer
from nltk.stem.lancaster import LancasterStemmer
from nltk.stem import SnowballStemmer
def obtain-tokens():
text-file = open("D:/Examples/Examples.txt", "r")
stem = text-file.read()
text-file.close()
tok = nltk.word-tokenize(stem)
return tok
def stemming(filtered):
stem=[]
for x in filtered:
stem.append(PorterStemmer().stem(x))
return stem
tok=obtain-tokens()
print("tokens is %s") stem-tokens= stemming(tok)
print(stem-tokens)
print("After stemming is %s"")
res=dict(zip(tok,stem-tokens))
print(res)
```

3.1 Lemmatization

Lemmatization is the process of determining the shape of the head (lexical appearance) of the word. The word formed after the lemmatization plays an important role. Through the morphy () method, the lemmatization process is performed in Wordnetlemmatizer. If the entered word is not found in the WordNet, it remains unchanged. Pos is formed part of the word category of the word entered in the argument. Let's look at the process of lemmatization in nltk [36, 38–40]: in the argument. Let's look at the process of lemmatization in nltk [36, 38–40]:

```
    import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer-output=WordNetLemmatizer()
print(lemmatizer-output.lemmatize('working'))
print(lemmatizer-output.lemmatize('working',pos='v'))
print(lemmatizer-output.lemmatize('works'))
```

WordNetLemmatizer the library generates Lemma through the Murphy () method using an information system (ontology) called WordNet Corps. If the lemma is not

formed, the word is returned in the initial (original) form. For example, the returned lemma unit forms for works: work. In the Uzbek language, the word lemmasi stemi of books will be a book. The following program code shows the difference between the stemming and lemmatization processes:

```
    import nltk
from nltk.stem import PorterStemmer
stemmer-output=PorterStemmer()
print(stemmer-output.stem(happiness)) from nltk.stem import WordNetLem-
matizer
lemmatizer-output=WordNetLemmatizer()
print(lemmatizer-output.lemmatize('happiness'))
```

```
    happi
happiness
```

In the previous code, happiness became happi as a result of the stemming process. Lemmatization can not find the core of the word happiness. Therefore, he returns the word happiness. The process of stemmatization in the Uzbek language is relatively easy, since in the Uzbek language lemmings often correspond to stem. But in the Uzbek language, too, cases of Flexion are flying. For example: in words such as: achievement, interrogative, the predicate is not an achievement, but an achievement, in the form of an interrogative. In such cases, the method of finding stem is used as above.

Polyglot – the software used to provide models called morphessor models, which are used to identify morphemes from tokens [41, 42]. The purpose of the Morpho project is to generate unmanaged processes for processing information. On the basis of the Morpho project, morphemes with the smallest unit of the syntax are created. Natural language morphemes play an important role in processing. Morphemes are used in automatic recognition and creation. With the help of polyglot's dictionaries, morphessor models were developed in 50000 tokens of different languages.

3.2 Morphological Analyzer

NIn the process of morphological analysis, the acquisition ofmatikmatic information is carried out taking into account the meanings of tokens based on attachments. Morphological analysis can be carried out in three ways:

1. morphology based on morpheme;
2. morphology based on lexeme;
3. ordinal-based morphology.

Morphological Analyzer is interpreted as a program that is responsible for analyzing the morphological composition of a particular token. Morphological analyzer analyzes the given token and form data such as Category, variety of meanings. To carry out morphological analysis on a token without a given space, a pyEnchant dictionary is

used. To determine the type of word, a set of rules is required. We can determine the word category by the following rules:

1. Morphological rules. Information about suffixes will help to determine the word category. For example, in English, the suffixes-ness and-ment are combined with nouns. So in English it is possible to determine its category, depending on the suffixes that make up the word chord. But in Uzbek it is not possible to determine the category of the word by this method. Therefore, the Uzbek language morpholexicon should be developed.
2. Syntactic rules. Contextual information helps to determine the word category. For example, if we find a word belonging to the category of nouns, then the syntactic rule is useful to determine whether the adjective comes before or after the noun category.
3. Semantic rules. Semantic rules are significant in determining the word category. For example, if it is determined that the word represents the place name, then it can be concluded that the noun belongs to the category of speech.
4. Open class (group). Every day a new word is added to the list of words in this group, their number increases. Words in the open class are usually words belonging to the category of nouns. Prepositions, in principle, belong to the closed class. For example, in the list there can be an infinite number of words.
5. Definition of word categories (POS, Part of Speech): a set of tags of word categories contains information that helps to determine the morphological feature. For example, oynadi Sor comes with a word belonging to the noun category in the third person unit.
6. Omorf package is licensed by GNU GPL version 3, it is used for many functions such as modeling, morphological analysis, rules-based machine translation, and data search, statistical machine translation, morphological segmentation, ontologies, spell checking and Correction.

3.3 Morphological Generator

Morphological generator is a program that performs the function of morphological generalization; it can be considered a dependent function of morphological analysis. Here the original word is formed if the description of the word according to the number, Category, core and other information is given. For example, **ozak=bormoq, gap bolagi = kesim, zamon=hozirgi and when a third person comes along with the subject of the unit, morphologically generator forms its becoming form.** There are many Python-based software applications that perform morphological analysis and generation:

1. **ParaMorfo:** It is a noun in Spanish and guarani, used for morphological formation and analysis of adjectives and verbs.
2. **HornMorpho:** It is used for morphological formation and analysis of nouns and verbs in the Oromo and Amharic languages, as well as Tigrinya verbs.
3. **AntiMorfo:** It is used for morphological creation and analysis of adjectives, verbs and nouns in the night language, as well as Spanish verbs.
4. **MorfoMelayu:** It is used for morphological analysis of words in the Malay language.
5. **Morph** morphological generator and analyzer for English.

6. **Morphy** morphological generator for German language, analyzer and POS tagger.
7. **Morphisto** it is a morphological generator and analyzer for the German language.
8. **Morfette** performs controlled learning (flexion morphology) of Spanish and French.

From the above feedback, it is possible to formulate the architecture of the text analyzer software (Fig. 4) as follows.

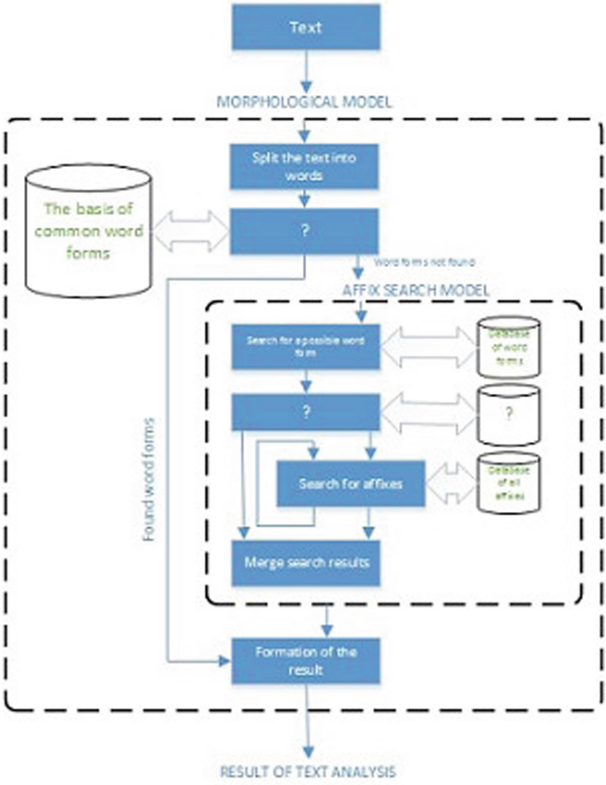


Fig. 4. Text Analyzer Software Architecture

4 Conclusion

There are many networks of computer linguistics. For processing (analysis) or generalization of the text, it is necessary to perform a number of linguistic actions on the given text. This article considered the implementation of stemming, lemmatization and morphological analysis and generalization activities with the help of NLTK package tools. Also, search engines and methods of their implementation were discussed.

References

1. Öztürel, A., Kayadelen, T., Demir¸sahin, I.: A syntactically expressive morphological analyzer for Turkish. In: FSMNLP 2019 14th International Conference on Finite-State Methods and Natural Language Processing, Proceedings (2019). https://doi.org/10.18653/v1/w19-3110
2. Jabbar, A., Iqbal, S., Akhunzada, A., Abbas, Q.: An improved Urdu stemming algorithm for text mining based on multi-step hybrid approach. J. Exp. Theor. Artif. Intell. **30**(5), 1–21 (2018). https://doi.org/10.1080/0952813X.2018.1467495
3. Szabó, G., Kovács, L.: Benchmarking morphological analyzers for the Hungarian language. Annales Mathematicae et Informaticae **49** (2018). https://doi.org/10.33039/ami.2018.05.001
4. Jurafsky, D.S., Martin, J.H., Kehler, A., Linden, K.V., Ward, N.: Speech and Langauge Processing, P. 950. Prentice Hall, Englewood Cliffs (2000)
5. Mohri, M.A.: Finite-state transducers in language and speech processing. Comput. Linguist. **23**, 269–311 (1997)
6. Alfred, V.A., Monika, S.L., Ravi, S., Djeffri, D.U.: Kompilyatory: prinsipy, texnologii i instrumentariy. OOO "I.D. Vilyamc", Per. sangl
7. Toldova, S.Yu.: Bonch-Osmolovskaya A.A. Avtomaticheskiy morfologicheskiy analiz. Fond znaniy "Lomonosov" (2011). www.lomonosov-fund.ru/enc/ru/encyclopedia:0127430
8. Sadykov, T., Kochkonbaeva, B.: Ob optimizatsii algoritma morfologicheskogo analiza. In: Shestaya Mejdunarodnaya konferensiya po kompyuternoy obrabotke tyurkskix yazykov. Turklang 2018. Trudy konferensii, Tashkent (2018)
9. Rodolfe, D.: Computational Linguistic Text Processing: Lexicon, Grammar, Parsing and Anaphora Resolution, p. 4–5. Nova Science Publishers, Inc., NewYork (2008)
10. Yermakov, A.: Morfologicheskiy analizator - osnova poiskovyx sistem. https://www.kv.by/archive/index2004154301.htm
11. Nojov, I.M.: Morfologicheskaya i sintaksicheskaya obrabotka teksta (modeli i programmy): dissertatsiya kand, p. 190. Moskva, nauk (2003)
12. Dybo, A.V., Sheymovich, A.V.: Avtomaticheskiy morfologicheskiy analiz dlya korpusov xakasskogo i drevnetyurkskogo yazykov. In: Nauchnoe obozrenie sayano-altaya retsenziruemyy nauchnyy jurnal Nomer **2**(08), 9–31 (2014)
13. Suleymanov, D.S., Gatiatullin, A.R.: Model tatarskoy affiksalnoy morfemy i yee realizatsiya, pp. 113–127. Intellekt. Yazyk. Kompyuter. – Vyp.4, Kazan, Seriya (1996)
14. Suleymanov, D.Sh., Gilmullin, R.A., Gataullin, R.R.: Morfologicheskiy analizator tatarskogo yazyka na osnove dvuxurovnevoy modeli morfologii. In: Pyataya Mejdunarodnaya konferensiya po kompyuternoy obrabotke tyurkskix yazykov TurkLang 2017, p. 327. Trudy konferensii. V 2-x tomax. T 2. Izdatelstvo Akademii nauk Respubliki Tatarstan, Kazan (2017)
15. Jeltov, P.V.: Razrabotka morfologicheskogo analizatora chuvashskogo yazyka. In: Pyataya Mejdunarodnaya konferensiya po kompyuternoy obrabotke tyurkskix yazykov TurkLang 2017. Trudy konferensii. V 2-x tomax. T 2, p. 327. Izdatelstvo Akademii nauk Respubliki Tatarstan, Kazan (2017)
16. Israilova, N.A., Bakasova, P.S.: Morfologicheskiy analizator kyrgyzskogo yazyka. In: Pyataya Mejdunarodnaya konferensiya po kompyuternoy obrabotke tyurkskix yazykov "TurkLang 2017". Trudy konferensii. V 2-x tomax. T 2, p. 327. Izdatelstvo Akademii nauk Respubliki Tatarstan, Kazan (2017)
17. Leontev, N.A.: Morfologicheskiy analizator yakutskogo yazyka. In: Shestaya Mejdunarodnaya konferensiya po kompyuternoy obrabotke tyurkskix yazykov "TurkLang-2018", vol. 320, pp. 276–279. (Trudy konferensii), Tashkent (2018)

18. Kukanova, V.V., Kadjiev, A.Y.: Algoritm raboty morfologicheskogo parsera kalmyskogo yazyka. In: V sbornike: Pismenoto nasledstvo i informatsionnite texnologii. El'Manuscript-2014 Materiali ot V mejdunarodna nauchnoy konferensii, pp. 116–119 (2014)
19. Orxun, M.: Computational analysis of uzbek nouns. In: Shestaya Mejdunarodnaya konferensiya po kompyuternoy obrabotke tyurkskix yazykov "TurkLang-2018", p. 320. Trudy konferensii, Tashkent (2018)