The pipeline processing of NLP

B. B. Elov[1*], *Sh. M.* Khamroeva[1], and *Z. Y.* Xusainova[1]

[1]Tashkent State University of Uzbek Language and Literature named after A.Navai, 100100 Tashkent, Uzbekistan

Abstract. The problem of NLP should be divided into several small parts and solved step by step. In this article, where NLP is necessary at every stage of solving the problem, all forms of text processing are considered. The step-by-step text processing is called a pipeline process in NLP. When creating any NLP model, the pipeline process is a sequence of steps that must be carried out. The planning and development of the text processing is considered as the starting point for the creation of any NLP project. This article discusses the steps involved in implementing a pipeline process and their role in solving NLP tasks. This article analyzed the most common preliminary processing steps on the NLP processing pipeline. All processing stages are pre-trained in various NLP libraries, identified as usable models. If necessary, additional, modified preprocessing steps can be developed depending on the given problem condition. One can determine how a particular initial processing stage serves a given NLP problem by many experimentations.

1 Introduction

Typically, the problem of NLP should be divided into several small parts and solved step by step. In this article, where NLP is necessary at every stage of solving the problem, all forms of text processing are considered. The step-by-step text processing is called a pipeline process in NLP [1-4]. When creating any NLP model, the pipeline process is a sequence of steps that must be carried out. The planning and development of the text processing is considered as the starting point for the creation of any NLP project. This article discusses the steps involved in implementing a pipeline process and their role in solving NLP tasks. Figure 1 below shows the main components of a common pipeline processing for developing a modern NLP system [5].

The main stages of the pipeline processing are as follows:

1. Data collection.
2. Text cleaning.
3. Initial processing.
4. Development of features.
5. Modeling.
6. Evaluation
7. Implementation.

[*] Corresponding author: elov@navoiy-uni.uz

8. Monitoring and model updating.

2 Main part

The first step in the development of any NLP system is data collection relevant to the given task. To develop a rule-based NLP system, it is necessary to have training (test) data to design and examine the rules. The data we receive is rarely “clean", and in the next step, the text cleaning procedure is performed. Once the text has been cleaned, the text data needs to be converted to canonical form.

This action is performed at the stage of preliminary processing [6-10]. After that, it is necessary to develop the necessary features to perform the NLP task. These features are converted into a comprehensible format by modeling algorithms. In the next step, modeling and evaluation are performed. In this step, one or more language models are created and they are compared using the appropriate evaluation indicators (metrics). It is necessary to implement this model after selecting the best one from among the developed models. And finally, we constantly monitor the performance of the model and, if necessary, need to improve its performance and update it. In the primary stages, a lot of time is spent on feature development, modeling, and evaluation [6, 7, 11-15].

Primary stages

As mentioned above, NLP software typically analyzes text by segmenting it into words (tokens) and sentences. Thus, any NLP pipeline processing needs to develop a system that correctly performs the segmentation of text into sentences (sentence segmentation) and subsequent segmentation of sentences into words (word tokenization).

Sentence segmentation

As a general rule, one can segment sentences by dividing text into sentences when periods and question marks appear. But there can be abbreviations, address forms (sh.k - according to, va b.-etc.) or ellipses (...) that can break the normal rule. The ability to distinct sentences and words using standard methods in existing NLP libraries is provided in Python. The following Python code shows how to use the sentence and word separator from the Natural Language Tool Kit (NLTK) library:

```
from nltk.tokenize import sent_tokenize, word_tokenize
mytext = "Typically, an NLP problem should be solved step by step, breaking it down into several small parts. This article examines all the forms of text processing required at each step of solving an NLP problem."
my_sentences = sent_tokenize(mytext)
```

Word tokenization

The main unit of the language is its vocabulary – lexicon. But what does the vocabulary (lexicon) contain? The term "word" has a broad meaning, and it is necessary to clarify this term in order to use it in a scientific manner. If we separate words from the text using separators (spaces, some punctuation marks, etc.), many tokens are generated in the text [16,17]. A token is any unit separated from the text by a boundary. There are 4 tokens in the sentence “Daraxt bir yerda koʻkaradi”: daraxt, bir, yerda, koʻkaradi. There are 4 tokens in the sentence “Halol mehnat yerda qolmas”: halol, mehnat, yerda, qolmas.

To turn sentences into words, like sentence tokenization, one can start with a simple rule of segmenting text into words based on the presence of punctuation marks. The NLTK library allows one to do this:

```
    for sentence in my_sentences:
print(sentence)
print(word_tokenize(sentence))
```

It should be remembered that most of the existing solutions (tokenizers) do not segment the given text into 100% correct tokens, and these algorithms are not perfect. For example, consider the following sentence:

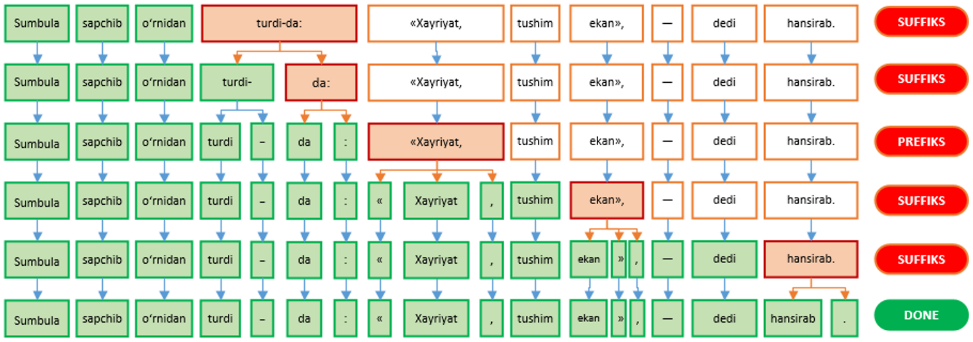
“Sumbula sapchib o‘rnidan turdi-da: «Xayriyat, tushim ekan», – dedi hansirab.”

If one proceed the above statement through the NLTK tokenizer, [o‘rnidan] [,] [turdi] [-] [da] [:] are defined as separate tokens. Also, if one wants to tokenize tweets, the tokenizer splits the hashtag into two tokens: the "#" sign and the sequence after it. In such cases, one may need to use a customized tokenizer that one creates (by one selves) for one’s purpose. Therefore, in the tokenizer, sometimes some elements of graphematic analysis must be performed:

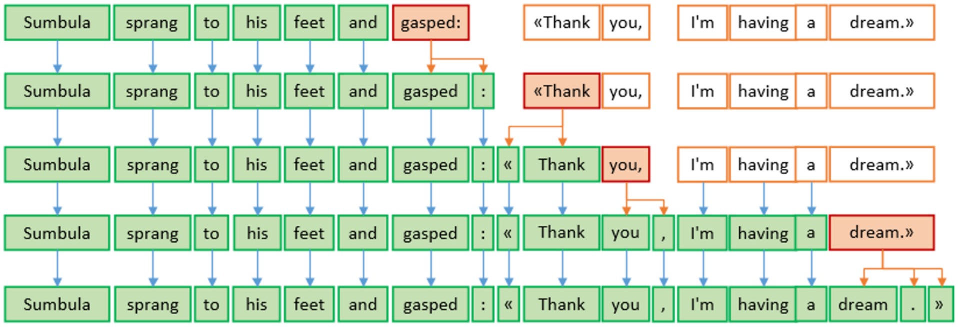
- sequence of letters;
- numbers;
- punctuation marks;
- hieroglyphs;
- separators;
- various graphic symbols.

Below is the tokenization process corresponding to 2 sentences in Uzbek and its version in http://uznatcorpara.uz/uz/Tokenizer program:

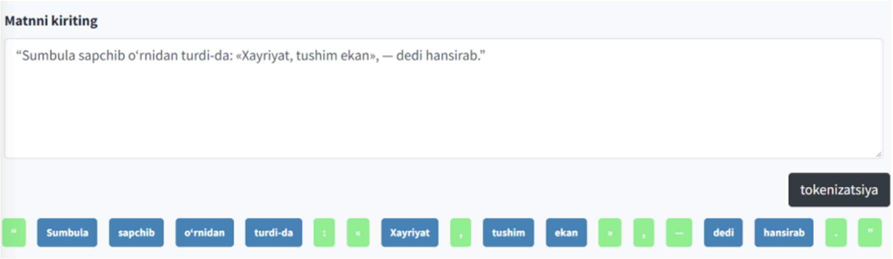
Sumbula sapchib o‘rnidan turdi-da: «Xayriyat, tushim ekan», – dedi hansirab - Sumbula sprang to his feet and gasped: "Thank you, I'm having a dream."


Sumbula | sapchib | o'rnidan | turdi-da: | «Xayriyat, | tushim | ekan», | – | dedi | hansirab. | SUFFIKS
Sumbula | sapchib | o'rnidan | turdi- | da: | «Xayriyat, | tushim | ekan», | – | dedi | hansirab. | SUFFIKS
Sumbula | sapchib | o'rnidan | turdi | - | da | : | «Xayriyat, | tushim | ekan», | – | dedi | hansirab. | PREFIKS
Sumbula | sapchib | o'rnidan | turdi | - | da | : | « | Xayriyat | , | tushim | ekan», | – | dedi | hansirab. | SUFFIKS
Sumbula | sapchib | o'rnidan | turdi | - | da | : | « | Xayriyat | , | tushim | ekan | » | , | – | dedi | hansirab. | SUFFIKS
Sumbula | sapchib | o'rnidan | turdi | - | da | : | « | Xayriyat | , | tushim | ekan | » | , | – | dedi | hansirab | . | DONE

English version tokenization process of sentences


Sumbula | sprang | to | his | feet | and | gasped: | «Thank | you, | I'm | having | a | dream.»
Sumbula | sprang | to | his | feet | and | gasped | : | «Thank | you, | I'm | having | a | dream.»
Sumbula | sprang | to | his | feet | and | gasped | : | « | Thank | you, | I'm | having | a | dream.»
Sumbula | sprang | to | his | feet | and | gasped | : | « | Thank | you | , | I'm | having | a | dream.»
Sumbula | sprang | to | his | feet | and | gasped | : | « | Thank | you | , | I'm | having | a | dream | . | »

Uzbek version tokenization process of sentences



Matnni kiriting

“Sumbula sapchib o‘rnidan turdi-da: «Xayriyat, tushim ekan», — dedi hansirab.”

“ Sumbula sapchib o‘rnidan turdi-da : « Xayriyat , tushim ekan » , — dedi hansirab . ”

Fig. 1. An example of the tokenization process

Steps to be taken regularly. Let's take a look at the initial processing steps that are performed regularly in the NLP pipeline processing. Let's say one developing software that classifies an article on a news site as one of politics, sports, business, and other topics. Let us have a program that divides a sentence into segments and tokens. In this case, one need to start thinking about what kind of data is useful for developing a tool for separating news into groups.

Some of the frequently used words in the Uzbek language, for example, albatta, ammo, asosan, aynan, balki, barcha, bilan, biroq are not considered necessary for this task. Because they don't mean anything by themselves to distinguish the four categories. Such words are called non-essential words and are usually removed from further analysis. However, there is no standard list of non-essential words for the Uzbek language. Some of the non-essential words in the Uzbek language developed by the authors are presented in the following Table 1.

Table 1. Common non-essential words in Uzbek language

unfortunately (afsuski)	Since (beri)	The most (eng)	With (Ila)	My (mening)	At the back (orqada)	shak-shubhasiz	ustida
apparently (aftidan)	With (bilan)	While (esa)	Let god (iloyim)	Like (Misli)	To the back (orqaga)	shekilli	ustidan
if(agar)	According to (binoan)	unfortunately (essiz)	I believe (ishonaman ki)	example (misoli)	Through (orqali)	shu	ustiga
on the contrary (aksincha)	But (biroq)	in return (evaziga)	Let it be (ishqilib)	during (mobaynida)	That (o‘sha)	shubhasiz	va
of course (albatta)	A bit (Biroz)	only (faqat)	In case of (jihatdan)	mobodo	Under (ostida)	shunchaki	vaqtda
already (allaqachon)	We (Biz)	If (gar)	Permissible (joiz)	On the event of(modomiki)	End (oxir)	shunday	xayriyat
in the end (alqissa)	Our (bizning)	though (garchand)	Very (juda)	Suitable (mos)	Itself (o‘zi)	shunga	xo‘sh
but (ammo)	Whole (bo‘yi)	although (garchi)	including (jumladan)	Neither .. nor …(na, na)	Myself (o‘zim)	shuning uchun	xolos
but (ammo-lekin)	Depends (bog‘liq)	as if (go‘yo)	Such as (Kabi)	Pity (nachora)	Ourselves (o‘zimiz)	shuningdek	xuddi
at least (aqalli)	Due to (bois)	supposedly (go‘yoki)	need (kerak)	Not only (nafaqat)	Yourself (o‘zingiz)	singari	xullas
Be about to (arafasida)	Other (boshqa)	sometimes (goh)	after (keyin)	Person (nafar)	Themselves (o‘zlari)	siz	xususan
Inter (aro)	This (bu)	like that (chunonchi	next (keyingi)	As a result (natijada)	Under (pastda)	sizniki	ya’ni

		)					
In fact (aslida)	These (bular)	Also (ham)	Who (kim)	The reason why (Negaki)	Under (pastga)	sizning	yana
Never (aslo)	From this (bundan)	Too (hamda)	Somebody (kimdir)	Whatever (Nimagaki)	in the beginning (payida)	tabiiyki	yanada
Mainly (asosan)	like this (bunday)	Every (hamma)	Whome (kimga)	Instead (o‘rniga)	When (qachonki)	tag‘in	yaxshi
Main (asosiy)	Whole (butun)	All (hammasi)	Many (ko‘p)	Between (o‘rtasida)	Till (qadar)	tahminan	yaxshiyam
First of all (avvalambor)	Approximately (chamasi)	Really (haqiqatda)	Most (ko‘plab)	That (o‘sha)	Looking at (Qarab)	tashqari	yo
Previous (avvalgi)	While (chog‘i)	Really (haqiqatdan)	Much (ko‘proq)	Self (o‘z)	To (qarata)	to‘g‘risi	yo‘qsa
Firstly (avvalo)	Because (chunki)	Every (har)	according to (ko‘ra)	Itself (o‘zi)	To (qaratilgan)	toki	yo‘q-yo‘q
after all (axir)	chunonchi	Anyway (har holda)	Much (ko‘proq)	Myself (o‘zim)	Again(qayta)	tomon	yo‘sinda
exactly (aynan)	Indeed (darhaqiqat)	Anyway (har qalay)	But (lekin)	Ourselves (o‘zimiz)	Instead of (qaytanga)	tufayli	yoki
the same (ayni)	Immediantly (darhol)	Even (hatto)	Must (lozim)	Yourself (o‘zingiz)	In short (qisqasi)	turli	yonida
Especially (ayniqsa)	Must (darkor)	Never (hech)	known (ma’lum)	Itself (o‘zini)	Let it be (qo‘yingki)	u	yonidan
For instance (aytaylik)	First of all (dastavval)	Etc (hokazo)	Whether (mabodo)	Themselves (o‘zlari)	Following (quyida)	uchun	yoniga
By the way (aytgancha)	While (davomida)	Only if (holda)	Here (mana)	Automatically (o‘z-o‘zidan)	Following (quyidagi)	They (ular)	yo‘q
By the way (aytganday)	Well (demak)	Now (hozir)	For example (masalan)	To be honest (ochig‘i)	In case (ravishda)	Them (ularni)	yoxud
Some (ba’zi)	Almost (deyarli)	At the moment (Hozirda)	Ok (mayli)	Related to (oid)	rostdan	ularning	yuqorida
Maybe (balki)	Let it be (deylik)	ichida	mazkur	olaylik	rosti	unda	yuqoriga
At the same time (baravarida)	Always (doim)	ichidan	mazmuni	oldiga	sana	unga	yuzasidan
All (barcha)	On this (doir)	ichiga	men	oldin	sari	uni	zero
All (barchasi)	Within (doirasida)	ichkarida	meni	orasida	sayin	uning	zeroki
Anyway (baribir)	Probably (ehtimol)	ichra	To my mind (menimcha)	Between (orasiga)	you think (seningcha)	This (ushbu)	zotan

Some NLP packages have lists of non-essential words (for some foreign languages), which in many cases can vary depending on the given problem.

Removing punctuation and/or numbers is a common step for many NLP problems such as text classification, data mining, and social media analysis. The following program code shows how to remove non-essential words, numbers, punctuation marks, and lowercase letters from a given set of English text:

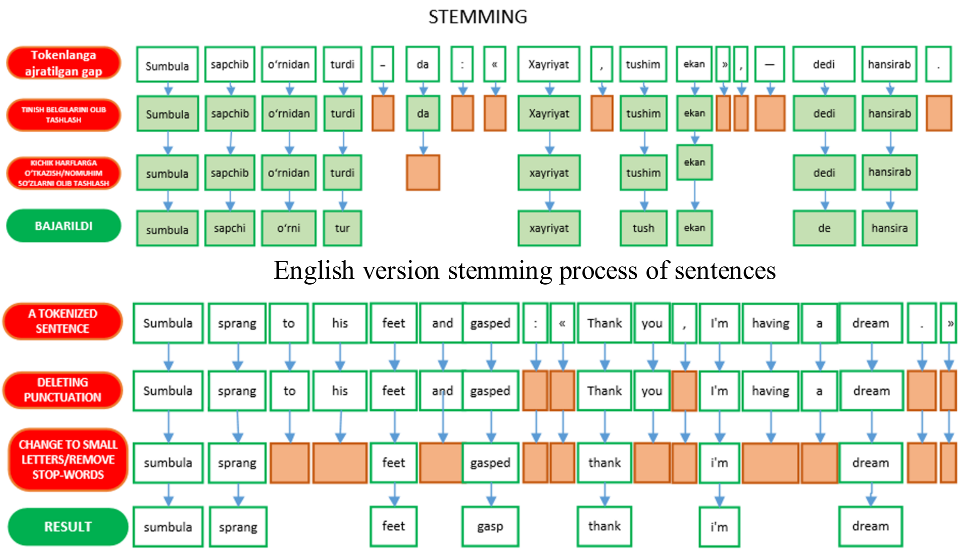
```
from uzcorpus import stopwords
From string import punctuation
def preprocess_corpus(texts):
   mystopwords = set(stopwords.words("uzbek"))
   def remove_stops_digits(tokens):
     return [token.lower() for token in tokens if token not in mystopwords
          not token.isdigit() and token not in punctuation]
   return [remove_stops_digits(word_tokenize(text)) for text in texts]
```

It should be noted that these four processes are not mandatory or required to be followed sequentially for all NLP problems. The function above shows our NLP project how to implement the text processing steps. Text data is primarily processed through this function. Commonly performed primarily processing steps that take into account word-level features are stemming and lemmatization.

3 Results

Stemming and lemmatization. In the process of stemming, suffixes are removed from the word and the word is reduced to some basic form (making the word form without suffixes).

For each token, there is an initial (or normal) form of it (called a lemma). In speech (text), this initial form is used in various grammatical forms (inflection may also occur). The following picture shows the stemming stages of two sentences.


STEMMING
Tokenlanga ajratilgan gap
Sumbula sapchib o'rnidan turdi – da : « Xayriyat , tushim ekan » , — dedi hansirab .
TINISH BELGILARINI OLIB TASHLASH
Sumbula sapchib o'rnidan turdi da Xayriyat tushim ekan dedi hansirab
KICHIK HARFLARGA O'TKAZISH/NOMUHIM SO'ZLARNI OLIB TASHLASH
sumbula sapchib o'rnidan turdi xayriyat tushim ekan dedi hansirab
BAJARILDI
sumbula sapchi o'rni tur xayriyat tush ekan de hansira
A TOKENIZED SENTENCE
Sumbula sprang to his feet and gasped : « Thank you , I'm having a dream . »
DELETING PUNCTUATION
Sumbula sprang to his feet and gasped Thank you I'm having a dream
CHANGE TO SMALL LETTERS/REMOVE STOP-WORDS
sumbula sprang feet gasped thank i'm dream
RESULT
sumbula sprang feet gasp thank i'm dream

English version stemming process of sentences

Uzbek version stemming process of sentences

Fig. 2. Examples of the stemming process

At this point, we need to clarify the term word form. A word form is a form of lemma (lexeme) used in speech. The lemma "Daraxt" can take various grammatical forms belonging to the noun group and form many word forms:

- daraxt, daraxtni, daraxtning, daraxtga, daraxtdan;
- daraxtlar, daraxtlarni, daraxtlarning, daraxtlarga, daraxtlardan;
- daraxtim, daraxting, daraxti, daraxtimiz, daraxtingiz, daraxtlari.

In the same order, other lemmas also appear as a word form when they are used in speech. The usual structure of the word structure in the Uzbek language is as follows:

"Stem + word-builder + lexical form + syntactic form"

The order and consistency of the placement of a grammatical device depends on its meaning and grammatical features, and is formed in the following sequence:

1) forming a new lexical meaning;
2) influencing the lexical meaning;
3) a tool that does not affect the lexical meaning, but connects the word is added.

The standard position of the suffix is sometimes broken:

opa-lar-im / opa-m-lar, ayt-di-ng-lar / ayt-di-lar-ing.

In the Uzbek language, additional morphemes are added consecutively after the stem. The rules of morphotactic determine the order in which morphemes and allomorphs are arranged in the word form. Based on these, we observe the connection possibilities of some morphemes in the Uzbek language in the table below:

TABLE 2. The morphemes set on the left side

morphemes	**left-hand positioning feature (+/-)**	Example in Uzbek	Example (translate to english)
-lar	+	bola+lar+dan	from children
-man	-	bor+a+man	I will go
-san	-	o‘qi+y+san	you read
-miz	-	yoz+a+miz	we write
-siz	-	ayt+ib+siz	you say
-ing	+	kel+ing	come
-im	+	kitob+im	my book
-imiz	+	daftar+imiz	our notebook
-si	+	uka+si	brother
-ni	-	ashula+ni	sing

From the table above, it can be seen that suffixes such as -lar, -im, -ing, -imiz, -si, -ni are added directly to the stem. It is added to the left side of another affix morpheme. The suffixes -man, -san, -miz, -siz cannot be added directly to the stem. Because before them, another morpheme (for example, a tense suffix) is added to the stem, and the next position is occupied by this group of suffixes, which have the characteristic of being located on the right side of affix morphemes. So, in the Uzbek language, the position of the stem and suffixes is minimally as follows:

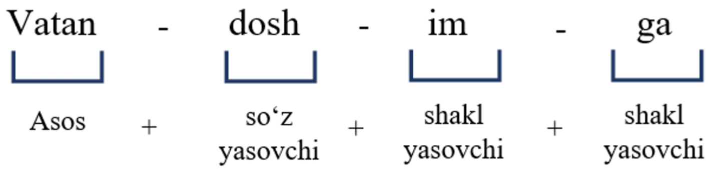


Fig. 3. position of the stem and suffixes

New words or word forms are formed by combining the smallest unit of language - morphs. Morphs are divided into stems (word bases) and suffixes:

- prefix (word-builder);
- a word-builder added after a word;
- syntactic form builder;

lexical form-building adverbs.

The stem means the main lexical meaning of the word. Suffixes give additional meaning to the word. Separating the word into morphemes is called morphemic analysis (Figure 5). For example:

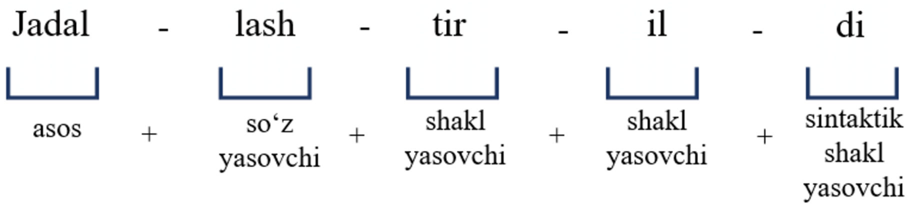


Fig. 4. morphemic analysis

In Figure 5 below, is presented the algorithm for determining the content of words:

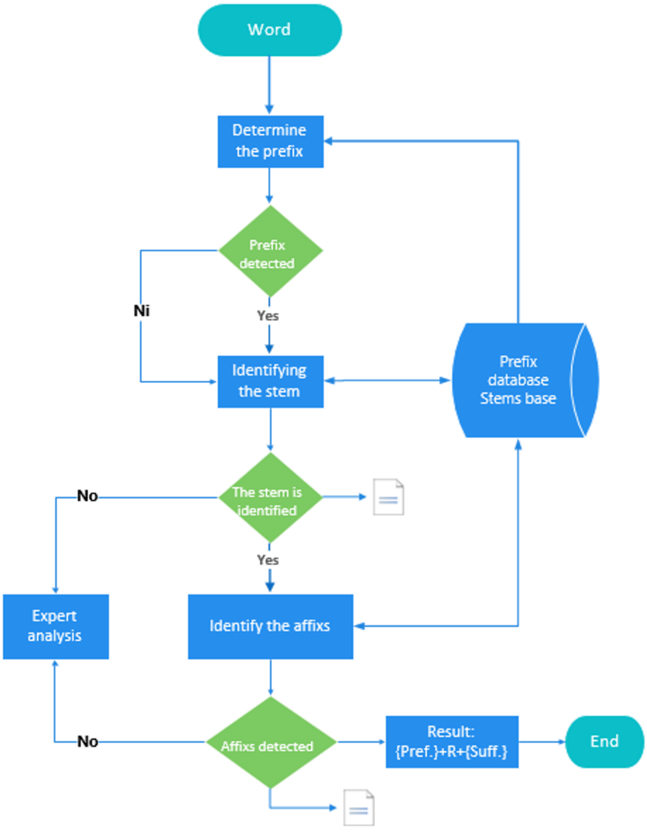


Fig. 5. Algorithm for determining word structure in Uzbek language

Lemmatization is the process of comparing all the different word forms of a word to a main word or lemma. Although this seems similar to the stemming process, they are actually different. The lemmatization process is shown below: Sumbula sprang to his feet and gasped: "Thank you, I'm having a dream."

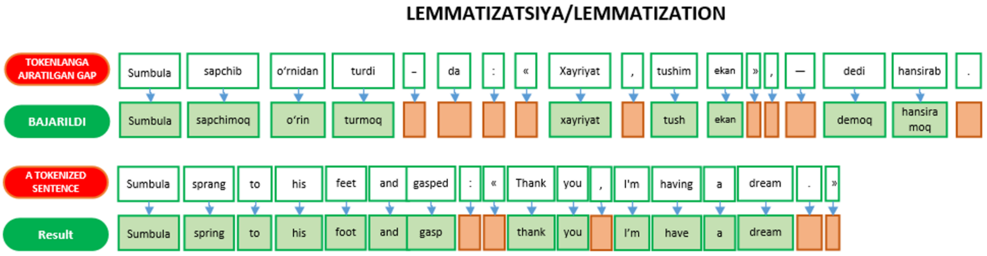
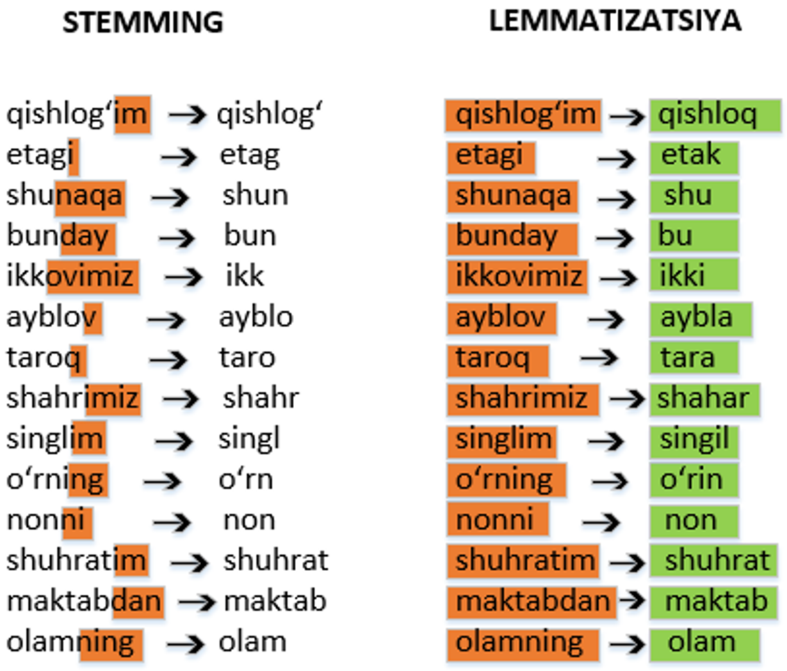


Fig. 6. Examples of the lemmatization process

It should be noted that stemming and lemmatization should be distinguished. Because often the results of stemming and lemmatization in the Uzbek language seem to be the same, but they are completely different processes: stemming is the process of removing the suffix from the stem, and lemmatization is the process of determining the dictionary variation of the stem. Below are examples of the analysis of the stemming and lemmatization process:



STEMMING			LEMMATIZATSIYA		
qishlog'im	→	qishlog'	qishlog'im	→	qishloq
etagi	→	etag	etagi	→	etak
shunaqa	→	shun	shunaqa	→	shu
bunday	→	bun	bunday	→	bu
ikkovimiz	→	ikk	ikkovimiz	→	ikki
ayblov	→	ayblo	ayblov	→	aybla
taroq	→	taro	taroq	→	tara
shahrimiz	→	shahr	shahrimiz	→	shahar
singlim	→	singl	singlim	→	singil
o'rning	→	o'rn	o'rning	→	o'rin
nonni	→	non	nonni	→	non
shuhratim	→	shuhrat	shuhratim	→	shuhrat
maktabdan	→	maktab	maktabdan	→	maktab
olamning	→	olam	olamning	→	olam

Fig. 7. The difference between stemming and lemmatization process.

Lemmatization requires more linguistic knowledge, because the modeling and development of efficient lemmatizators for Uzbek language texts is still an open problem in NLP research.

Since lemmatization involves a certain amount of linguistic analysis of the word and its context, it is more time-consuming than the stemming process, and it is usually used only when necessary. Which lemmatizator or stemmer to use for the initial processing stages of the NLP pipeline processing is chosen according to the condition of the given problem.

Not all steps to remove non-essential words, numbers, punctuation, and lowercase letters from a given text are always necessary. For example, if one removes numbers and punctuation marks from text, the removal may not be very important at first. However, one usually replaces uppercase letters with lowercase letters before performing the stemming process on the text.

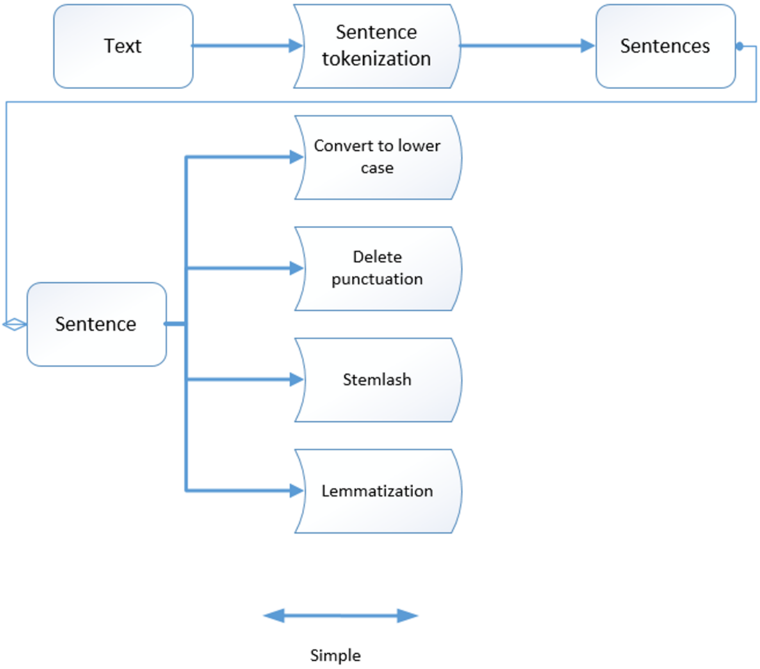


Fig. 8. Common primary processing steps for a text fragment

One does not remove lexemes and lowercase letters from the text before the lemmatization process. Because in order to get the lemma, one needs to know the part of speech, and this requires that all tokens in the sentence are intact. Once one have a clear understanding of how to process data, it's a good idea to prepare a step-by-step list of primary processing steps to be taken. Examples of the Uzbek language morphoanalyzer developed by the authors (http://uznatcorpara.uz/) to tokenization, stemming and lemmatization steps in the NLP pipeline processing and the ER model in the database are presented below:

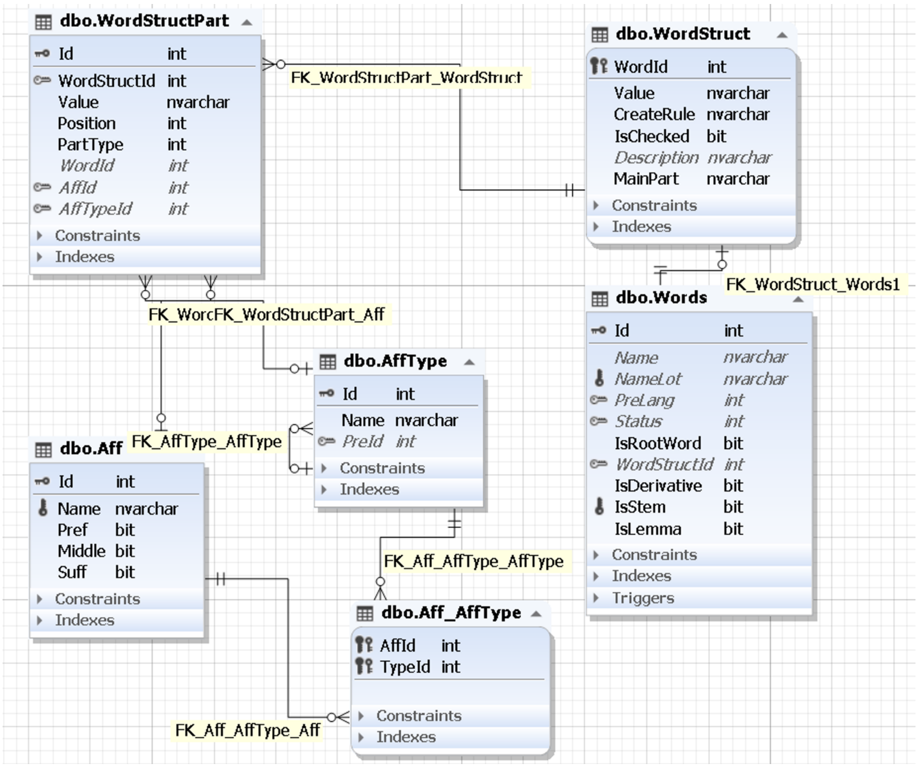
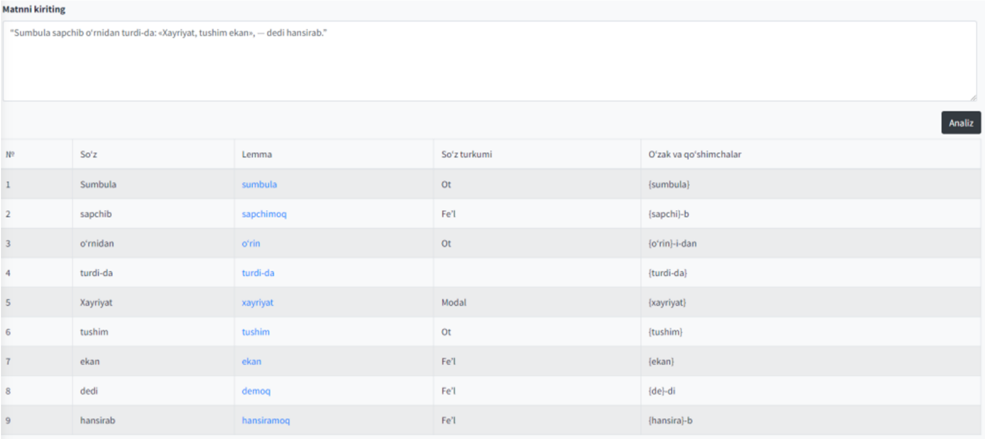


Fig. 9. NLP pipeline processing ER model


Matnni kiriting

"Sumbula sapchib o'rnidan turdi-da: «Xayriyat, tushim ekan», — dedi hansirab."

№	So'z	Lemma	So'z turkumi	O'zak va qo'shimchalar
1	Sumbula	sumbula	Ot	{sumbula}
2	sapchib	sapchimoq	Fe'l	{sapchi}-b
3	o'rnidan	o'rin	Ot	{o'rin}-i-dan
4	turdi-da	turdi-da		{turdi-da}
5	Xayriyat	xayriyat	Modal	{xayriyat}
6	tushim	tushim	Ot	{tushim}
7	ekan	ekan	Fe'l	{ekan}
8	dedi	demoq	Fe'l	{de}-di
9	hansirab	hansiramoq	Fe'l	{hansira}-b

Fig. 10. Morphoanalyzer of the Uzbek language (http://uznatcorpara.uz/)

Additional stages of primary processing. A few common primary preprocessing steps in the NLP pipeline processing have been covered above. Although the essence of the texts is not clearly indicated, it is assumed to work with plain English text. Below are given some additional primary processing steps.

Text normalization. Let's look at the issue of identifying news in social media posts. Social media text is quite different from the language used in newspapers. Words can be written in different ways, for example, in abbreviated forms, phone numbers can be written in different formats, names are sometimes written in lowercase letters, etc. When developing NLP tools to work with such data, we need to create a canonical form of the text that covers all changes. This process is called text normalization. Some common steps to normalize text are converting text to all lowercase or uppercase, converting numbers to text (e.g. 9 to nine), expanding abbreviations, etc. A simple method of text normalization is provided in the Spacy package.

Identifying the language. Most of the web content is written in languages other than English. For example, let's say one was asked to collect all the reviews of product on the internet. Analyzing various e-commerce websites, when one starts scanning product pages, one come across a few non-English reviews. Since the main part of the NLP pipeline processing is built with language-specific tools, is it necessary to make regular changes in our NLP pipeline processing that is waiting for English text? In such cases, language identification is performed as the first step in the NLP pipeline processing. One can use NLP packages like Polyglot for language identification. After this operation, the next steps of the NLP pipeline processing can be observed to be language specific.

Many people around the world speak more than one language in their daily lives. Thus, they use several languages in their posts on social networks. As an example of code mixing, we can see the phrase Singlish (Singapore slang + English language) in LDC in Figure 5. One phrase contains words from Tamil, English, Malay and three Chinese languages. Code mixing refers to this phenomenon of switching between languages. When people use more than one language in their writing, they often write words in those languages in Latin script, with English spelling. So, along with the English text, words in other languages are also written. This process is known as transliteration. Both of these phenomena are common in multilingual communities and should be taken into account during the primary text processing.

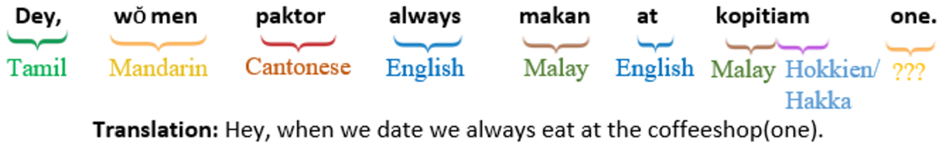


Fig. 11. Post in Singlish

The general primary processing steps have been discussed above. Although this list is not complete, we hope it gives one an idea of the different primary processing methods that may be required depending on the nature of the data set.

4 Extended processing

We will consider the issue of developing a system for identifying the names of individuals and organizations in a collection of one million documents in the company. The common steps of text processing that we have discussed earlier may not be appropriate in this context. Identifying names requires us to implement POS tags. Because identifying the appropriate proper nouns can be useful in determining the names of individuals and organizations. How do we do POS tags during the primary processing stage of the project? Primary-trained and easy-to-use POS taggers are used in NLP libraries like NLTK, spaCy, and Parsey McParseface Tagger. One typically don't need to develop its own POS tagging solutions. It is important to note that for the same primary processing step, there may be differences between the results from different NLP libraries.

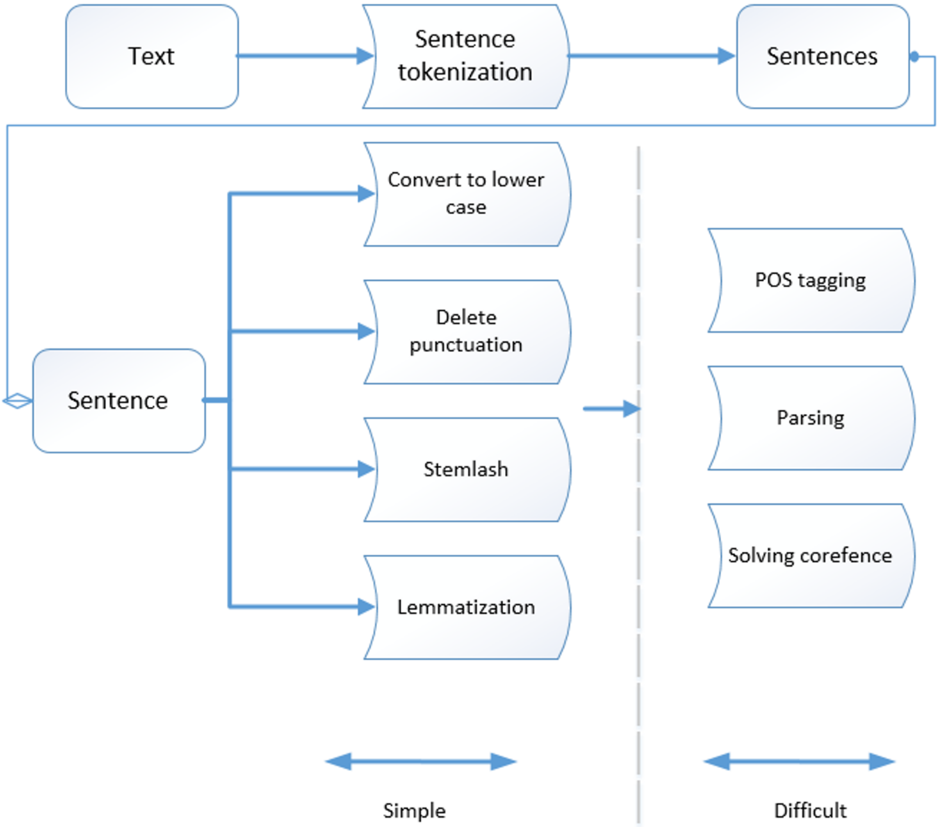


Fig. 12. Extended processing steps for a text fragment

This is due to differences in implementation and algorithms between different libraries. Which library (or libraries) to use in an NLP project depends on the context of the given problem. Let's look at a slightly different problem: in addition to identifying the names of individuals and organizations in our company's collection of millions of documents, we are also asked to determine whether a particular individual and organization are somehow related. To do this, we need to develop a method for identifying patterns that indicate the "relationship" between two persons in a sentence. This requires us to have some form of syntactic representation of the sentence.

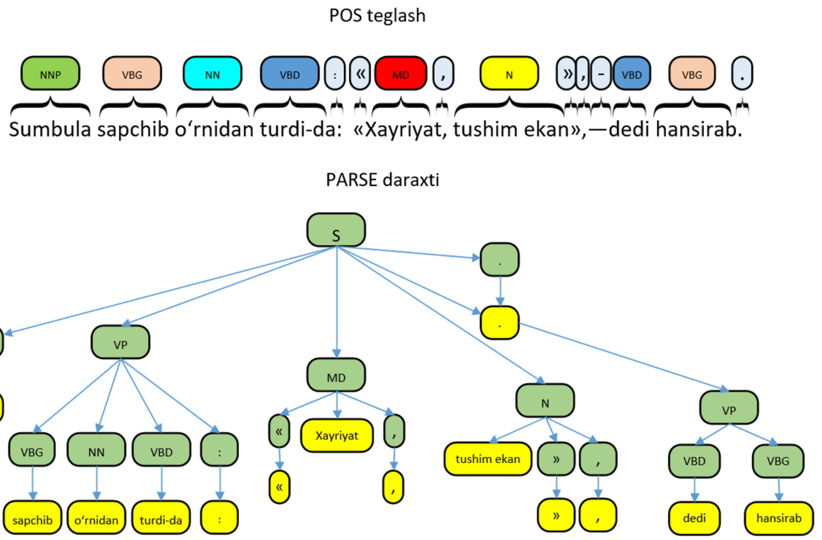


Fig. 13. An example of POS tagging and sentence structure in the NLP pipeline processing

In addition, we need to outline a method to define and bind multiple references to an object.

5 Conclusion

This article analyzed the most common primary processing steps in the NLP pipeline processing. All processing steps are defined as primary-trained, usable models in various NLP libraries. If necessary, additional, customized primary processing steps can be developed depending on the given problem condition. One can determine how a particular initial processing stage serves a given NLP problem by many experimentations. In the morphoanalyzer of the Uzbek language developed by the scientists of Tashkent State University of Uzbek Language and Literature named after Alisher Navoi, tokenization, stemming and lemmatization stages of the NLP pipeline processing were developed and implemented.

References

1. M. A. Saloot, D. N. Pham, *Real-time Text Stream Processing: A Dynamic and Distributed NLP Pipeline*, ACM International Conference Proceeding Series (2021) https://doi.org/10.1145/3459104.3459198
2. G. Becquin, End-to-end NLP Pipelines in Rust (2020) https://doi.org/10.18653/v1/2020.nlposs-1.4
3. N. Peng, F. Ferraro, M. Yu, N. Andrews, J. DeYoung, M. Thomas, M. R. Gormley, T. Wolfe, C. Harman, B. van Durme, M. Dredze, *A concrete Chinese NLP pipeline. NAACL-HLT 2015 - 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Demonstrations, Proceedings (2015) https://doi.org/10.3115/v1/n15-3018
4. H. Noji, Y. Miyao, Jigg: A framework for an easy natural language processing pipeline. 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 System Demonstrations (2016) https://doi.org/10.18653/v1/p16-4018

5. S. Vajjala, B. Majumder, A. Gupta, H. Surana, Practical Natural Language Processing. A Comprehensive Guide to Building Real-World NLP Systems, 455 (2020)
6. E. B. Botir, X. I. Axmedova, Business Process Modeling That Distinguishes Homonymy Within Three Parts of Speechs in Uzbek Language, International conference on information science and communications technologies application, trends and opportunities (IEEE - UBMK - VII. Uluslararası Bilgisayar Bilimleri ve Mühendisliği Konferansı), Ankara (2022)
7. B. Elov, Sh. Hamraeva, X. Axmedova, *Methods for creating a morphological analyse*, 14th International Conference on Intellegent Human Computer Interaction. 19-23 October, Tashkent (2022)
8. B. Elov, Sh. Hamroyeva, D. Elova, Morfologik analizatorni yaratish usullari, O ‘zbekiston: til va madaniyat. Amaliy filologiya masalalari, **5(1)**, 67-87 (2022)
9. B. R. Menliev, Sh. M. Khamroeva, Structure and units of the morphoanalyzer of the Uzbek language, Computer linguistics and vychislitelnye ontologii. Vypusk 5 (Trudy XXIV Mejdunarodnoy ob'edinennoy nauchchestsii "Internet i sovremennoe obshchestvo", IMS-2021, Sbornik nauchnyx trudov), Saint-Petersburg, University ITMO, 82 (2021)
10. B. B. Elov, Text generation in Uzbek using N-gram language models, Computational linguistics: problems, solutions and perspectives, Collection of international scientific and practical conference. Electronic publication, ebook, Tashkent (2022)
11. E. Soysal, J. Wang, M. Jiang, Y. Wu, S. Pakhomov, H. Liu, H. Xu, CLAMP - a toolkit for efficiently building customized clinical natural language processing pipelines, Journal of the American Medical Informatics Association, **25(3)** (2018) https://doi.org/10.1093/jamia/ocx132
12. I. Tenney, D. Das, E. Pavlick, *BERT rediscovers the classical NLP pipeline. ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics*, Proceedings of the Conference (2020) https://doi.org/10.18653/v1/p19-1452
13. G. Attardi, *DeepNL: A deep learning NLP pipeline. 1st Workshop on Vector Space Modeling for Natural Language Processing*, VS 2015 at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT (2015) https://doi.org/10.3115/v1/w15-1515
14. S. Koeva, N. Obreshkov, M. Yalamov, *Natural language processing pipeline to annotate bulgarian legislative data. LREC 2020 - 12th International Conference on Language Resources and Evaluation*, Conference Proceedings (2020)
15. W. de Vries, A. van Cranenburgh, M. Nissim, What’s so special about BERT’s layers? A closer look at the NLP pipeline in monolingual and multilingual models, Findings of the Association for Computational Linguistics Findings of ACL: EMNLP (2020) https://doi.org/10.18653/v1/2020.findings-emnlp.389
16. B. Elov, Tabiiy tilni qayta ishlash (nlp)da spacy modulidan foydalanish. Science and innovative development, Tashkent, **4**, 41-55 (2022)
17. Z. Y. Xusainova, NLP: tokenizatsiya, stemming, lemmatizatsiya va nutq qismlarini teglash. O‘zbek amaliy filologiyasi istiqbollari, Respublika ilmiy-amaliy konferensiya to‘plami. Elektron nashr, Toshkent: ToshDO‘TAU, 159-163 (2022)