The pipeline processing of NLP

B. B. Elov, Sh. M. Khamroeva, Z. Y. Xusainova

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 [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. 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.
7. Implementation.

* Corresponding author: elov@navoiy-uni.uz

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).
Monitoring and model updating.

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. 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.

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 norm. 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:

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

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. There are 4 tokens in the sentence "Daraxt bir yerda ko'karadi": daraxt, bir, yerda, ko'karadi. 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:

```python
for sentence in my_sentences:
    print(sentence)
    print(word_tokenize(sentence))
```
It should be remembered that most of the existing solutions 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.
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>
<thead>
<tr>
<th>Table 1. Common non-essential words in Uzbek language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unfortunately (afsuski)</td>
</tr>
<tr>
<td>Since (beri)</td>
</tr>
<tr>
<td>The most (eng)</td>
</tr>
<tr>
<td>With (Ila)</td>
</tr>
<tr>
<td>My (mening)</td>
</tr>
<tr>
<td>At the back (orqada)</td>
</tr>
<tr>
<td>Shak-shubha siz ustida</td>
</tr>
<tr>
<td>Apparently (aftidan)</td>
</tr>
<tr>
<td>With (bilan)</td>
</tr>
<tr>
<td>While (esa)</td>
</tr>
<tr>
<td>Let god (iloyim)</td>
</tr>
<tr>
<td>Like (Misli)</td>
</tr>
<tr>
<td>To the back (orqaga)</td>
</tr>
<tr>
<td>Shak-shubha siz ustida</td>
</tr>
<tr>
<td>If (agar)</td>
</tr>
<tr>
<td>According to (binoan)</td>
</tr>
<tr>
<td>Unfortunately (essiz)</td>
</tr>
<tr>
<td>I believe (ishonaman ki)</td>
</tr>
<tr>
<td>Example (misoli)</td>
</tr>
<tr>
<td>Through (orqali)</td>
</tr>
<tr>
<td>Shu ustiga</td>
</tr>
<tr>
<td>On the contrary (aksincha)</td>
</tr>
<tr>
<td>But (biroq)</td>
</tr>
<tr>
<td>In return (evaziga)</td>
</tr>
<tr>
<td>Let it be (ishqilib)</td>
</tr>
<tr>
<td>During (mobaynida)</td>
</tr>
<tr>
<td>That (o'sha)</td>
</tr>
<tr>
<td>Shubha siz va vaqtda</td>
</tr>
<tr>
<td>Of course (albatta)</td>
</tr>
<tr>
<td>A bit (Biroz)</td>
</tr>
<tr>
<td>Only (faqat)</td>
</tr>
<tr>
<td>In case of (jihatdan)</td>
</tr>
<tr>
<td>Mobodo (ostida)</td>
</tr>
<tr>
<td>Shunchaki vaqtda</td>
</tr>
<tr>
<td>Already (allaqacho n)</td>
</tr>
<tr>
<td>We (Biz)</td>
</tr>
<tr>
<td>If (gar)</td>
</tr>
<tr>
<td>Permissible (joiz)</td>
</tr>
<tr>
<td>On the event (modomi ki)</td>
</tr>
<tr>
<td>End (oxir)</td>
</tr>
<tr>
<td>Shunda xayriyat xo'sh</td>
</tr>
<tr>
<td>But (ammo)</td>
</tr>
<tr>
<td>Whole (bo'yi)</td>
</tr>
<tr>
<td>Although (garchi)</td>
</tr>
<tr>
<td>Including (jumladan)</td>
</tr>
<tr>
<td>Neither .. nor …(na, na)</td>
</tr>
<tr>
<td>Myself (o'zim)</td>
</tr>
<tr>
<td>chunonchi next keyingi</td>
</tr>
<tr>
<td>As a result (natijada)</td>
</tr>
<tr>
<td>Under (pastda)</td>
</tr>
<tr>
<td>sizniki yani xayriyat xolos</td>
</tr>
<tr>
<td>But (ammo-lekin)</td>
</tr>
<tr>
<td>Depends (bog'liq)</td>
</tr>
<tr>
<td>as if (go'yo)</td>
</tr>
<tr>
<td>Such as (Kabi)</td>
</tr>
<tr>
<td>Pity (nachora)</td>
</tr>
<tr>
<td>Ourselves (o'zimiz)</td>
</tr>
<tr>
<td>xo'sh xullas</td>
</tr>
<tr>
<td>at least (aqalli)</td>
</tr>
<tr>
<td>Due to (bois)</td>
</tr>
<tr>
<td>Supposedly (go'yoki)</td>
</tr>
<tr>
<td>need (kerak)</td>
</tr>
<tr>
<td>Not only (nafaqat)</td>
</tr>
<tr>
<td>Yourself (o'zingiz)</td>
</tr>
<tr>
<td>singari xa'niya</td>
</tr>
<tr>
<td>Be about to (arafasida)</td>
</tr>
<tr>
<td>Other (boshqa)</td>
</tr>
<tr>
<td>sometimes (goh)</td>
</tr>
<tr>
<td>after (keyin)</td>
</tr>
<tr>
<td>Person (nafar)</td>
</tr>
<tr>
<td>Themself (o'zlari)</td>
</tr>
<tr>
<td>ya'ni</td>
</tr>
</tbody>
</table>

Table 1. Common non-essential words in Uzbek language
In fact, these explanations are also needed.

Who (kim) is the reason (Negaki) behind this (pastga)?

Never (aslo) from this (bundan) too (hamda) somebody (kimdir) whatever (Nimagaki) in the beginning (payida) natural (tabiiyki) to the yada (yanada).

Mainly (asosan) like this (bunday) every (hamma) who (kimga) instead (o'rniga) looking at (Qarab) the tag'in (yaxshi) in the (yanada) on the (yaxshi) you.

Mainly (asosiy) whole (butun) all (hammasi) many (ko'p) between (o'rtasida) till (qadar) estimate (tahmin) and (an) well (yo).

First of all (avvalamb or) approximately (chamasi) really (haqiqatda) most (ko'plab) that (o'sha) looking at (Qarab) the (tashqar) i (yo).

Previous (avvalgi) really (haqiqatda) much (ko'proq) myself (o'z) to (qaratilgan) toki (yaxshi) - yo'q-

After all (axir) anyway (har hol is) much (ko'proq) myself (o'zim) again (qayta) to mon (yo'sin) also (yo'qsa) the same (ayni)

Indeed (darhaq) anyway (har qalay) but (lekin) ourselves (o'zimiz) instead of (qaytang) to tufayli (yoki) the same (ayni)

Especially (ayniqsa) must (darkor) never (hech) known (ma'lum) itself (o'zini) let it be (qo'yingki) to that (yoni) n yonida (yoniga).

By the way (aytgancha) while (davomida) only if (holda) here (mana) automatic (o'z-o'zida) following (quyidagi) they (ular) yo'q

By the way (aytganha) well (demak) now (hozir) for example (masalan) to be honest (ochig'i) in case (ravishda) them (ularni) yoxu yonida yonida (yuqor va yo'qi).

Some (ba'zi) almost (deyarli) at the moment (Hozirda) ok (mayli) related to (oid) rostdan ularnin yuqor da yuqor ga.

Maybe (balki) let it be (deylik) ichida mazkurb olaylik rosti unda yuqor ga.

At the same time (baravarida) always (doim) ichidan mazmuni oldiga sana u yuzasi dan.

All (barcha) on this (doir) ichiga men oldin sari uni zero.

All (barchasi) within (doirasida) ichkarida meni orasida sayin uning yuziga.

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...
3 Results

![Diagram of stemming process](image)

**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:

<table>
<thead>
<tr>
<th>morphemes</th>
<th>left-hand positioning feature (+/-)</th>
<th>Example in Uzbek</th>
<th>Example (translate to English)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-lar</td>
<td>-lar</td>
<td>bola+lar+dan</td>
<td>from children</td>
</tr>
<tr>
<td>-man</td>
<td>-man</td>
<td>a+man</td>
<td>I will go</td>
</tr>
<tr>
<td>-san</td>
<td>-san</td>
<td>y+san</td>
<td>you read</td>
</tr>
<tr>
<td>-miz</td>
<td>-miz</td>
<td>a+miz</td>
<td>we write</td>
</tr>
<tr>
<td>-siz</td>
<td>-siz</td>
<td>i+y+siz</td>
<td>you say</td>
</tr>
<tr>
<td>-ing</td>
<td>-ing</td>
<td>kel+ing</td>
<td>come</td>
</tr>
<tr>
<td>-im</td>
<td>-im</td>
<td>kitob+im</td>
<td>my book</td>
</tr>
<tr>
<td>-imiz</td>
<td>-imiz</td>
<td>daftar+imiz</td>
<td>our notebook</td>
</tr>
<tr>
<td>-si</td>
<td>-si</td>
<td>uka+si</td>
<td>brother</td>
</tr>
<tr>
<td>-ni</td>
<td>-ni</td>
<td>ashula+ni</td>
<td>sing</td>
</tr>
</tbody>
</table>

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](image)

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

<table>
<thead>
<tr>
<th>Vatan -dosh -im -ga</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asos + so‘z yasovchi + shakl yasovchi + shakl yasovchi</td>
</tr>
</tbody>
</table>
- 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.

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:

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.

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://uznatorparad.uz/](http://uznatorparad.uz/)) to tokenization, stemming and lemmatization steps in the NLP pipeline processing and the ER model in the database are presented below:
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 a different format, and so on.
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](image)

Translation: Hey, when we date we always eat at the coffeeshop (one).

4 Extended processing
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.
5 Conclusion

This article analyzed the most common primary processing steps in the NLP pipeline. 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


12. I. Tenney, D. Das, E. Pavlick, BERT rediscovers the classical NLP pipeline. ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics


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

