**- Progress Report (15.11.2022) -**

# Turkish NLP Preprocessing Toolbox

**Göktuğ Öcal and Tunga Güngör**
Boğaziçi University
Department of Computer Engineering
34342 Bebek, Istanbul, Turkey
{goktug.ocal, gungort}@boun.edu.tr

## Abstract

Natural Language Processing (NLP) is a branch of Artificial Intelligence which is the ability of understanding text and speech as possible as a human being can. There are various applications of NLP in the business such as sentiment analysis, text classification, chatbots, speech recognition, spam detection etc. Preprocessing is the prior step in the NLP techniques which consists of tokenization, sentence splitting, normalization, stemming, lemmatization, stopword removal, lower casing and maybe more. This project provides couple of text preprocessing steps focused on Turkish language and words with a useful toolbox.

## 1 Introduction

Artificial Intelligence (AI) uses computers and other devices to simulate how the human mind makes decisions and solves problems [1]. Artificial intelligence has demonstrated its ability to tackle problems that are simple for humans to carry out but challenging for humans to articulate in a formal manner. These are issues that we intuitively and automatically solve, including identifying spoken words or faces in pictures. That is why, AI is a thriving field with many practical applications and active research topics (Bengio et al., 2017).

Natural Language Processing is a field of AI which is capturing, analyzing, understanding, and deriving meaning from human language in a smart and useful way in order to enabling computers to understand text or spoken words as the way humans understand. More precisely, NLP gives linguistic skills to computers. This skills are added to computers with different applications such as search engines, autocomplete, language translation, chatbots, voice assistants, grammar checking, email filtering, sentiment analysis and social media monitoring.

Various techniques are built and used in order to use for NLP tasks. There are statistical language models such as Markov Models, N-Gram Models and also there are state-of-art ML and AI based neural language models such as RNN models, BERT models, K-Mean Models. Regardless from the which language model is used in the task, text preprocessing is an essential step in preparing the text data for better analysis. Text preprocessing is the first step in the pipeline of NLP, with potential impact in its final result. Text data contains noise in various forms like emotions, punctuation, text in a different case. Text preprocessing is the process of bringing the text into a form that is predictable and analyzable for a specific task. Therefore, this prior step of the NLP pipeline is important to conduct successfully in order to more accurate and semantic results.

## 2 The Project

The project requires to design and implement a preprocessing toolbox which is going to work on the Turkish language with its grammer rules. The desired toolbox has to have the following preprocessing functionalities:

- Tokenization: The tokenizer should take into account special entities like URLs, hashtags, e-mails, MWEs (multi-word expressions), etc. A rule based and a ML based method should be applicable in that step.

[1] https://www.ibm.com/cloud/learn/what-is-artificial-intelligence

Table 1: Current status of the project

| | **Status** | |
|---|---|---|
| **Tokenization** | Regex rule-based algorithm is implemented | ML-based algorithm is **not implemented yet** |
| **Sentence Splitting** | Regex rule-based algorithm is implemented | ML-based algorithm is **not implemented yet** |
| **Normalization** | Implemented with Edit Distance and a Frequency Table | |
| **Stemming** | Implemented with Regex-based suffix removal and Stem list | |
| **Stopword Removal** | Implemented with Stopword list | |

- Sentence splitting: The sentence splitter should be able to detect different types of end-of-sentence markers. It should also handle complex cases such as the occurrence of end-of-sentence markers within quotations. A rule based and a ML based method should be applicable in that step.
- Normalization: The normalizer should be able to work on general domain (e.g. news domain), rather than being specific to some particular domains.
- Stemming: The stemmer should take into account all inflectional suffixes and a wide range of derivational suffixes.
- Stopword elimination: This operation should be for the general domain; not for the stopwords of a particular domain.

## 3 Current State of the Project - The Toolbox

Our approach while building the toolbox is designing a simple preprocessing library that can be used easily in the NLP pipeline. The toolbox is coded with Python and only a few additional libraries have been used (such as Regex library). All the components of the toolbox can be run independently for each task and those preprocessing functionalities can be executed in a pipeline one by one. Current state of the required functionalities is shown in Table 3.

**Components of Toolbox**

- **Tokenization**
  Tokenization, which is a common task in NLP, is a way of separating a piece of text into smaller units called tokens which can be either words, characters, or subwords. Whitespace/unigram tokenization is the most popular tokenization technique. By separating the words from the whitespace, the entire text is divided into words in this procedure. Regular expression tokenization is another variety of tokenization. In which the tokens are obtained using a regular expression pattern. The tokenization might be done at the character level, word level, or even sentence level.

  In the toolbox, tokenization is done with regular expression rules since there are multiple token types such as word tokens, financial representations, dates, abbreviations etc. In the Table **??**, regular expressions patterns that have been used have given. With different types of patterns each token can be splitted properly and types of the tokens can be assigned such as EOS token, date token etc.

  For a given sample text:

  *Saçma ve Gereksiz Bir Yazı.*
  *Bakkaldan 5 TL'lik 2 çikola-*
  *ta al. 12.02.2018 tarihinde saat tam 15:45'te yap-*

Table 2: Used Regex patterns for tokenization

| | Patterns |
|---|---|
| **Financial** | (?:[\$€]\d+(\.\d\d\d)*(,\d\d)?)<br>(?:\d+(\.\d\d\d)*\s?(try\|türk l[iI]rası\|tl\|\|l[iI]ra\|avro\|euro\|dolar))<br>(?:(%\|‰\|yüzde\|b[iI]nde)\s?\d+(,\d+)?)<br>(?:\d+(,\d+)?(%\|‰)) |
| **Date** | (?:\b(0[1-9]\|1[0-9]\|2[0-9]\|3[0-1])\s<br>([eoşmnhtak][kcuaiIeğyr][abrsyzmuliI][a-zıü]{1,4})<br>(\s\d{2,4})?)<br>(?:\b(0[1-9]\|1[0-9]\|2[0-9]\|3[01])[/.-](0[1-9]\|1[0-2])[/.-]\d{2,4})<br>(\d{1,4})(?= yılı) |
| **Time** | (?:(0?[0-9]\|1[0-9]\|[0-2][0-4])[.:][0-5][0-9])<br>(?:(0?[0-9]\|1[0-2])([.:][0-5][0-9])?\s?(am\|pm\|a\.m\.\|p\.m\.)) |
| **Phone** | (?:(\+9\|009)?0?\s?(5[0-5][0-9])\s?\d{3}\s?\d{2}\s?\d{2})<br>(?:(\+9\|009)?0?\s?([234][0-9][0-9])\s?\d{3}\s?\d{2}\s?\d{2}) |
| **Web** | (?:[a-z0-9_\-\.]+)@([a-z0-9_\-\.]+)\.([a-z]{2,5})<br>(?:(https?://)?(www\.)?[a-z0-9]+\.[a-z]{3,6}[.a-z0-9\?=\-\+%&_/]*) |
| **Number** | (?:(\d+)([,.]\d+)?)<br>(?:(\d{1,3})([,.]\d{3})*([,.]\d+)?) |
| **Proper Noun Abbr.** | ((?<=\s)([a-zçğıIöşü]\.)) |
| **End of sentence** | (\?\.\.)\|(!\.\.)\|(\.\.\.)\|(\?)\|(!)\|(:)\|(\.)\|(\.\") |
| **Words** | ([0-9a-zÀ-ž]+([’']\[a-zçğıöşü]+)?) |
| **Suffix** | (?<=')([a-zÀ-ž]+) |
| **New line** | \n |
| **White Space** | \s |
| **Punctuation** | [^\w\s] |

*malıyız bu işi. Tamam mı? Benimle goktugocal41@gmail.com adresinden iletişime geçebilirsin.*

the tokenized output would be like that:

*[’Saçma’, ’ ’, ’ve’, ’ ’, ’Gereksiz’, ’ ’, ’Bir’, ’ ’, ’Yazı’, ’.’, ”, ’Bakkaldan’, ’ ’, ’5 TL’, ”’”, ’lik’, ’ ’, ’2’, ’ ’, ’çikolata’, ”, ’ ’, ’al’, ’.’, ’ ’, ’12.02.2018’, ’ ’, ’tarihinde’, ’ ’, ’saat’, ’ ’, ’tam’, ’ ’, ’15:45’, ”’”, ’te’, ’ ’, ’yapmalıyız’, ”, ’ ’, ’bu’, ’ ’, ’işi’, ’.’, ’ ’, ’Tamam’, ’ ’, ’mı’, ’?’, ’ ’, ’Benimle’, ’ ’, ’goktugocal41@gmail.com’, ’ ’, ”, ’adresinden’, ’ ’, ’iletişime’, ’ ’, ’geçebilirsin’, ’.’]*

OVerall the tokenization is working well but for other input types it should be investigated.

- **Sentence Splitting**
  Text can be divided into sentences using the ’.’ or ’/n’ characters, making this chore of doing so relatively simple. In free text data, however, this pattern is inconsistent, and writers are permitted to omit a line break or use ”.” in the incorrect context.

  With a proper tokenization technique as described in the previous step, the sentence splitting can be done easily with assigned token types. Each ”EOS” token would point a sentence end in a text. In the sentence splitting function of the toolbox, the function takes the text and runs Tokenizer at first then captures the token types and tokens. Then investigates ”EOS” tokens ans separates sentences.

  For the same input given in previous section the token types would be like this:

  *[[’word’, ’space’, ’word’, ’space’, ’word’, ’space’, ’word’, ’space’, ’word’, ’eos’], [’word’, ’space’, ’financial’, ’punch’, ’suffix’, ’space’, ’number’, ’space’, ’word’, ’space’,*

*'word', 'eos'], ['space', 'date', 'space', 'word', 'space', 'word', 'space', 'word', 'space', 'time', 'punch', 'suffix', 'space', 'word', 'space', 'word', 'space', 'word', 'eos'], ['space', 'word', 'space', 'word', 'eos'], ['space', 'word', 'space', 'web', 'space', 'word', 'space', 'word', 'space', 'word', 'eos']]*

therefore, the sentences would be splitted as follows:

*['Saçma ve Gereksiz Bir Yazı.', "Bakkaldan 5 TL'lik 2 çikolata al.", " 12.02.2018 tarihinde saat tam 15:45'te yapmalıyız bu işi.", ' Tamam mı?', ' Benimle goktugocal41@gmail.com adresinden iletişime geçebilirsin.']*

- **Normalization**
  Generally normalization is the attempting to reduce randomness of text, bringing it closer to a predefined standard. This helps us to reduce the amount of different information that the computer has to deal with, and therefore improves efficiency.

  In the toolbox, normalization is conducted with deriving variations of the word and checking whether the derived word is existing in a predefined frequency table. Frequency table is gathered from web[2]. The derivation process has a simple approach as implementing all possible vowels between two consonant in the word. In the second step, all the possible derivations are checked whether they are in the frequency table and frequency of the match. If there is a match also Levenshtein Distance is measured as an additional parameter and all the matches append to a candidate list. In the decision step, it is required to select a word from candidate list. To accomplish this, the list is sorted and the word with smaller Levenshtein Distance and higher frequency is selected.

  For the same input given with a small pertubations, tokens would be like this:

  *['Sçma', ' ', 've', ' ', 'Gereksiz', ' ', 'Bir', ' ', 'Yzı', '.', '', 'Bakkaldan', ' ', '5 TL', "'", 'lik', ' ', '2', ' ', 'çikolata', '', ' ', 'al', '.', ' ', '12.02.2018', ' ', 'tarihinde', ' ', 'saat', ' ', 'tam', ' ', '15:45', "'", 'te', ' ', 'yapmalıyız', '', ' ', 'bu', ' ', 'işi', '.', ' ', 'Tamam', ' ', 'mı', '?', ' ', 'Bnimle', ' ', 'goktugocal41@gmail.com', ' ', '', 'adresinden', ' ', 'iletişime', ' ', 'geçebilirsin', '.']*

  the normalized output would be like that:

  *['saçma', ' ', 've', ' ', 'gereksiz', ' ', 'bir', ' ', 'yazı', '.', '', 'bakkaldan', ' ', '5 TL', "'", 'elik', ' ', '2', ' ', 'çikolata', '', ' ', 'al', '.', ' ', '12.02.2018', ' ', 'tarihinde', ' ', 'saat', ' ', 'tam', ' ', '15:45', "'", 'te', ' ', 'yapmalıyız', '', ' ', 'bu', ' ', 'işi', '.', ' ', 'tamam', ' ', 'mi', '?', ' ', 'banimle', ' ', 'goktugocal41@gmail.com', ' ', '', 'adresinden', ' ', 'iletişime', ' ', 'geçebilirsin', '.']*

  In result, the Normalizer is worked well in normalizing pertubated words but there are some problems such as converting "bnimle" to "banimle". The correct word should be "benimle". That behavior should be investigated in further works.

- **Stemming**
  Stemming is the process of converting a word to its most general form, or stem. This helps in reducing the size of our vocabulary. An approach to stemming is removing the last few characters of a given word, to obtain a shorter form. Generally the removing few characters may work but for a agglutinative language like Turkish it is more suitable to remove suffixes from the word in order to retrieve the root of the word. Deciding whether the leftover piece of the word after suffix removal is a root or not is an important step to avoid from overstemming or understemming. To that purpose the BOUN Turkish Treebank(Turk et al., 2020) is used as stem list. Another requirement in stemming is the list of suffixes. A list of nominal and derivational suffixes is used to find them in a word with regular expression engine.

  For the same input given in previous sections the stems types would be like this:

[2]https://github.com/brolin59/trnlp

*['saçma', ' ', 've', ' ', 'gereksiz', ' ', 'bir', ' ', 'yazı', '.', '', 'bakkal', ' ', '5 tl', "'", 'e', ' ', '2', ' ', 'çikolata', '', ' ', 'al', '.', ' ', '12.02.2018', ' ', 'tarih', ' ', 'saat', ' ', 'ta', ' ', '15:45', "'", 'te', ' ', 'yap', '', ' ', 'bu', ' ', 'iş', '.', ' ', 'tamam', ' ', 'mi', '?', ' ', 'be', ' ', 'goktugocal41@gmail.com', ' ', '', 'adres', ' ', 'iletişim', ' ', 'geçebilirsin', '.']*

The stemming performance overall is seems like well but there are some errors such as "gereksiz". The stem of the word is "gerek" but the stemming could not find that. It is observed that when the list of suffixes is extended the stemming becomes more accurate. In the further work the more suffixes are tried to be added to the list.

- **Stopword Removal**
  Stop-words are commonly used words in a language. Stop-words are removed from the text so that we can concentrate on more important words and prevent stop-words from being analyzed. In onther languages there are plenty of stop-words can be addressed but in Turkish, most of the stop-words are conjunctions. The method used in stop-word removal is simply looking at whether the word in the stop-word table or not in order to make a decision to eliminate that word.

  The list for stop-words in in english is gathered from the web[3]. After that the simple removal process is applied in this functionality of the toolbox. For the same input given in the previous section and after the stemming is done, the output of the stop-word removal is gives the following output:

  *['saçma', 'gereksiz', 'yazı', '.', '', 'bakkal', '5 tl', "'", '2', 'çikolata', '', 'al', '.', '12.02.2018', 'tarih', 'saat', '15:45', "'", 'te', 'yap', '', 'iş', '.', 'mi', '?', 'ba', 'goktugocal41@gmail.com', '', 'adres', 'iletişim', 'geçebilirsin', '.']*

## 4 Further work in the project

- The project is requires additional functionalities in the toolbox. First of all, machine learningb-based approaches for tokenization and sentence splitting must be added to the toolbox. Character-based Naïve Bayes and logistic regression models are going to be used in those preprocessing steps.
- Implemented preprocessing algorithm are going to be tested with the BOUN Turkish Treebank and the required fixes are going to conducted.
- Investigated works and studies in the community and the literature are going to be put together and added to the final project report.

[3] https://github.com/ahmetax/trstop/blob/master/dosyalar/turkce-stop-words

## References

Yoshua Bengio, Ian Goodfellow, and Aaron Courville. 2017. *Deep learning*, volume 1. MIT press Cambridge, MA, USA.

Utku Turk, Furkan Atmaca, and Şaziye Betül Özateş and Gözde Berk and Seyyit Talha Bedir and Abdullatif Köksal and Balkız Öztürk Başaran and Tunga Güngör and Arzucan Özgür. 2020. Resources for turkish dependency parsing: Introducing the boun treebank and the boat annotation tool.