Karahan Şahin & Büşra Marşan

GROUP 10 PROGRESS REPORT

Before we started our application project, we did a brief literature review to guide the process. One of the resources we found particularly useful was Liu (2018) which discusses a normalization task for Swedish. Since Swedish has relatively richer morphology than English, they needed to bear in mind the particularities of the language and adapt a more localization-based approach that makes use of phonological and morphological features of the language. This particular feature of this study attracted our interest and we aimed to create a preprocessing toolkit that pays attention to the linguistic typology and morphophonological characteristics of Turkish.

Our progress report covers each step of the normalization process individually. The resources we used and relevant references can be found at the end of this document.

1. Tokenization

For this step, we created an extensive Multi Word Expression list using Turkish PARSEME 1.2 corpus[1] (see Sak et al. (2019) for a detailed discussion) which consists of 22,311 sentences (LVC.full: 3582, VID: 4139, MVC: 5). We used the PASEME 1.2 corpus as a lemma list with a broad coverage since this corpus contains different inflectional forms of each word. A total of 2874 multiword expressions are included in the corpus but voicing alternation is present (causative, passive etc.). For our purposes within the framework of this application process, we don’t need such forms. That is why we are removing these forms from the multi word expression list.

[1] https://gitlab.com/parseme/sharedtask-data/-/tree/master/1.2/TR

Instead of including every single voicing alternation form as individual items, we opted for encoding voicing alternation in the final consonants of the verbs thus employing a syntactic and phonological representation. Thus, “e[d/t]” is the representation of “etmek” and “edilmek” in our list.

In addition, we made use of the in-class discussion regarding tokenization and used the following features:

#Rule 1: Put white-space around unambiguous punctuation (? ( ) ; etc.)

# Rule 2: Put white-space around commas that aren’t inside numbers

# Rule 3: Put white-space after single quotes not preceded by a letter

# Rule 4: Put white-space after clitics not followed by a letter

2. Sentence Splitting

One of the main challenges for sentence splitting was abbreviations. In order to tackle this issue, we referred to Mikheev (2002). The current status of our solution is a somewhat primitive one and we aim to improve it in the near future. For this purpose, we are taking a closer look at Stevenson & Gaizauskas (2000) to derive inspiration.

For sentence splitting task, we opted for using Multinomial Naive Bayes, and for tokenization, we decided to use Logistic Regression. The main motivation behind our choice was an in-class discussion where it was mentioned that Logistic Regression was a better fit when dealing with large quantities of data.

As also mentioned in Mikheev (2002), full stops in abbreviations were an issue. To deal with the abbreviations, we exported some data from TDK's web page[2][3] and came up with an extensive list. Then we created regular expressions to cover the entire paradigm. Using these regular expressions, we extract relevant features for punctuation marks, mainly full stops. Then using these features, we go through the classification task to decide whether we are at a sentence boundary or not. For instance, is_abbr feature is referred to in order to decide whether there is an abbreviation before the punctuation. If there is not an instance of an abbreviation before the punctuation mark, that instance is classified as a sentence boundary, otherwise it is classified as a "not-sentence-boundary".

Features	Description
is_capital	Whether the following token starts with capital letter
is_space	Whether the punctuation followed by space character
is_number	Whether the preceded or followed by a number
is_alpha	Whether preceded by a one alphabetic character
is_acronym	Whether preceded by an acronym
length_prev	Total number of preceded characters between another non-alphanumeric character

3. Normalization

[2] https://www.tdk.gov.tr/icerik/yazim-kurallari/kisaltmalar-dizini/
[3] https://tdk.gov.tr/wp-content/uploads/2019/01/Kısaltmalar_Dizini.pdf

We are still working on this step. Our goal is to incorporate abbreviation normalization (such as “btw” -> “by the way”) and substitution of contractions (such as “Ali’den” -> “Ali”, “den”).

4. Stemming

Due to the morphologically complex nature of Turkish, creating a high performance stemmer is a serious challenge. The thinly veiled distinction between nominal morphology and verbal morphology makes this challenge a gargantuan one: The same person and number agreement markers are stacked in the final position on both verbs and nouns. That is why disambiguating nouns and verbal predicates is a very difficult task. In order to fully capture such particularities of Turkish, we took a linguistics-based approach to create layers in the algorithm. Referring to the thorough discussion of morpheme stacking in Göksel (2015) *(see Figure 1),* we created an algorithm that trims each word starting from the right hand side. This algorithm refers to the linear order of the morphemes to correctly stem potentially confusing words like “hayatım”. This word can be confused as a verbal predicate, thus stemmer can yield “haya” as a verbal root after a slicing like the following:

-ım -> Person Agr.

-t -> Causative Marker

haya- -> Verbal root

To yield the correct stem, the algorithm must “know” that this word is a noun, hence it cannot bear accusative morpheme. The decision of treating the word as a noun or a verbal predicate is made in accordance with the following heuristics:

```
If Mod -DIR is present, it is a verbal predicate.
        Follow the morpheme hierarchy for verbal morphology.
```

If the final morpheme is a person agreement marker, look to the left to make a decision.

If the left hand side bears nominal morphology (such as -lAr), treat the word as a noun and slice morphemes while referring to nominal morphology.

If the left hand side bears,

TAM-II

Follow the morpheme hierarchy for verbal morphology.

TAM I

Follow the morpheme hierarchy for verbal morphology.

(…)

	Voice	Negation	Modality	TAM I	TAM II	Agr.	Mod
Verb +	-Il, -DIr, -In, -Iş	-mA	-(y)abil	-DI, -mIş, -Iyor, -(y)AcAk, -Ar/Ir, -sA, -mAlı, -(y)A, -mAkta	idi, imiş, ise	person	-DIR

Figure 1

We used UD morphology to extract all morphological features and create their corresponding regular expressions. In addition, we make use of a derivational suffix list. The list of derivational suffixes along with morphological features and their corresponding regular expressions serve as a resource for the stemming task. Currently, our stemming algorithm is in the development phase in line with our insights and decisions discussed above.

5. Stop-word Elimination

For the static stop word detection, we made use of NLTK's Turkish stop words (nltk.stopwords("turkish")) list. We are still working on the dynamic stop word detection approach.

6. Some Additional Notes

Before we conclude our progress report, we'd like to discuss some finer points. Our Naive Bayes Model is a multi-purpose one. In other words, it is not "specified" for the task of sentence splitting. Moreover, it utilizes Pandas dataframe. To build and refine this model, we referred to the in-class discussions and lecture notes.

We plan to use KeNet[4] UD style dependency corpus (see Bakay et al. (2021) for a detailed discussion of Turkish WordNet KeNet, and de Marneffe et al. (2021) for a contemporary discussion of Universal Dependencies) to train our models. The reason behind this choice is the fact that KeNet dependency corpus is already tokenized, and the data provided there is annotated. We plan to format it as a dataset with markers at the end of each token (something along the lines of (token<end>)) before using.

Currently we are using the following libraries:

1. regex
2. pandas
3. numpy
4. sklearn
5. nltk

[4] https://universaldependencies.org/treebanks/tr_kenet/index.html

REFERENCES

Bakay, Ö., Ergelen, Ö., Sarmış, E., Yıldırım, S., Arıcan, B. N., Kocabalcıoğlu, A., ... & Yıldız, O. T. (2021, January). Turkish wordnet kenet. In *Proceedings of the 11th global wordnet conference* (pp. 166-174).

de Marneffe, M. C., Manning, C. D., Nivre, J., & Zeman, D. (2021). Universal dependencies. *Computational linguistics*, *47*(2), 255-308.

Liu, Y. (2018). A Pipeline for Automatic Lexical Normalization of Swedish Student Writings.

Mikheev, A. (2002). Periods, capitalized words, etc. *Computational Linguistics*, *28*(3), 289-318.

Sak, H., Güngör, T., & Saraçlar, M. (2011). Resources for Turkish morphological processing. *Language resources and evaluation*, *45*(2), 249-261.

Stevenson, M., & Gaizauskas, R. (2000, April). Experiments on sentence boundary detection. In *Sixth Applied Natural Language Processing Conference* (pp. 84-89).

Taylan, E. (2015). 2.5. Verbal Inflection. In *The phonology and morphology of Turkish*. essay, Boğaziçi Üniversitesi.