Bachelors thesis
Multilingual social media data tokenization and corpus creation

Ivan Petrov
klims008@gmail.com

Abstract

Multilingual social media data presents a big challenge for NLP tasks and tokenization is the first hurdle. In order to tackle this challenge, a basis for these experiments is needed, as well as consideration of already existing approaches. This project aims to begin creating such a basis and looks into some of the ways in which existing solutions fail. I compile existing data into a more homogeneous dataset and evaluate four existing tokenizers. This is done using a scoring system on a corpus of eight languages, and using qualitative analysis in order to dive into the lower-scoring instances. I found that both rule-based and trained tokenizers tested have similar shortcomings, however the trained model was affected to a lesser extent. The main issues were the ability to distinguish emoticons, identify typos and non-whitespaced tokens, usage of symbols from the same script in different languages/cases and inconsistency in the normalization of the data. Furthermore, there is an additional challenge in using a trained model, as the origin of some of the wrong tokenization may not be clear.

GitHubRepo: [1]

1 Introduction

While tokenization of standard text can be considered a solved task for many languages, tokenization of social media and other quick and informal text-based forms of communication is not. The difficulty stems from the frequent use of non-standard words and shorthands, as well as code-switching, unusual punctuation and typos. In addition, multilinguality adds even further challenge, adding a lot more variations of text and ways of how it is encoded. In order to investigate multilingual social media data tokenization in an efficient manner there needs to be a concrete and methodical way to approach it. This project aims to provide a basis for this by compiling a dataset which can be used for such experiments and also provide some insight into where pre-existing tokenizers fall short in this task.

2 Related works

2.1 Multilexnorm

The backbone of this project is the data from the normalization task Multilexnorm [13]. The task data consisted of tokenized text and normalized text pairs. For the project we consider only the tokenized text and accept that as the ”gold standard”. The task page also included links to the papers from which the data was compiled, which in turn allowed me to find the original ”untokenized” data for some of the languages.

3 Datasets

3.1 Overview

Below you can find the statistics for the datasets used

	English	Danish
Name	Tweebank v2	DaN+
Tokens	55,607	3430
Texts	3,550	213

Table 1: English and Danish dataset stats

	Italian	Slovene
Name	UD_Italian-PoSTWITA	Janes-Tag
Tokens	119334	75276
Texts	6712	2958

Table 2: Italian and Slovene dataset stats

3.2 Serbian, Croatian and Slovenian

These three datasets come from clarin.si. The Serbian [7] and the Croatian[8] datasets come from

	Spanish	Croatian
Name	Tweet_Norm 2013	NormTagNER-hr
Tokens	13824	89855
Texts	1100	3871

Table 3: Spanish and Croatian dataset stats

	Serbian	Irish
Name	NormTagNER-sr	UD_Irish-TwittIrish
Tokens	92271	47790
Texts	3748	2596

Table 4: Serbian and Irish dataset stats

ReLDI network and the Slovene data comes from Janes-Tag[5], All three are manually anotated and are ”meant as a gold-standard training and testing dataset for tokenisation, sentence segmentation, word normalisation, morphosyntactic tagging, lemmatisation and named entity recognition of non-standard (Croatian, Slovene, Serbian).” Sizes given in overview.

3.3 Universal Dependencies

Two of the datasets used were taken from Universal Dependencies Treebanks[9]: UD_Italian-PoSTWITA and UD_Irish-TwittIrish. They were both in CONNLU format.

The English dataset is a collection of English tweets from Universal Dependencies which is repackaged as Tweebank v2.[6]

3.4 Spanish

The Spanish dataset came from a normalization competition Tweet-Norm [2] at the SEPLN 2013 conference. This was the only dataset that required me to slightly normalize the original text as the tokens were all lowercase.

3.5 Danish

The Danish dataset[11] came from various places and was the messiest by far. The data was collected from the Universal Dependencies treebank, Reddit, Twitter and Arto, which was a danish social media platform. I was able to recover 213 out of the 900 texts from it.

4 Methodology

4.1 Matching algorithm

The method of finding the matching ”detokenized” text from the existing tokenized text varied due to the data being in different forms. The algorithmic approach consisted of a number of steps. Firstly, convert the raw untokenized text into a string. Then for each of the tweets/texts, find all instances of the first token in the text with regex. Take the next 560 characters as a contender for a match and calculate a score for it by seeing how many of the tokens are in the text. The algorithm awards extra points if the tokens are found sequentially and follow each other in the text. Highest score wins and some further logic exists to cover some of the corner cases. Another function further cuts off the the text based on the last token. The text is then saved as a pandas dataframe in the format of [Original],[Tokenized] where the tokenized is saved as a string with the separator ’omega_ts’. I opted to use a longer separator due to the texts varying a lot and the combination of any symbols may be present in the texts, as well as special symbols being generally annoying to work with in regex. Some of the datasets were already in a CONNLU or CONNLUP format, which allowed me to forego the matching algorithm and grab both the text and the tokens from the same file.

4.2 Further data processing

After creating the matched files, I reformatted them all into a CONLLU type format and split them into 60/40 training/evaluation splits.

5 Experiments

5.1 Chosen metrics

For the evaluation I used recall, but slightly modified. Since the ”predicted” tokenized list could have a different size than the ”true” tokenized list, I created a list of all ones and then also created a dictionary of all tokens in the true list together with their counts. Then iterating throught the list of predicted tokens, I would see if the token was in the ”true” list, keeping in mind the number of the same token present in the text. I would then end up with a true list full of ones and a predicted list to compare against. I chose to use a non weighted average as that would reflect and highlight the failure cases better.

5.2 Evaluation

After acquiring the scores for each language using the tokenization methods, I used a jupyter notebook to manually look at cases with lower than average for all languages and find the common failure points for each tokenizer of each language. My findings are presented below. I decided to go with manual evaluation due to the complexity of the task.

6 Tokenizers

6.1 NLTK

6.1.1 Quantitative Results

	es	sr	sl	hr
Recall	0.893	0.970	0.961	0.958

Table 5: NLTK recall scores for the first half of the languages

	da	en	ir	it
Recall	0.941	0.894	0.975	0.961

Table 6: NLTK recall scores for the second half of the languages

6.1.2 Qualitative results

The NLTK tweet tokenizer[3] is the most common on the list of the ones used. The two lowest scores were achieved for English and Spanish. Upon examination into where the English tokenization fails I found three main points of failure. The first one was the concatenation of punctuation and special symbols. The NLTK tokenizer recognizes the more common combinations such as ":)" and ":')", however there are many other variations which it just isn't equipped to deal with, such as ">.<" or ":L". The library also put repetitions into separate tokens whereas the true tokenization concatenated them into one, i.e. "!", "!" versus "!!".

The second point of failure were contractions. They are very common in English and the NLTK library handled them differently from the "true" tokenization. The library just took the word as a whole token, i.e. "didn't" whereas the true tokenization made two separate tokens, "did" and "n't".

The last major difference was to do with the URLs in the text. The true tokenization employed a anonymization technique, which replaced with URLS with "URLX" where X is a number unique to that specific URL. It then used that as a whole token. The NLTK library however split the URL and the number into two separate tokens.

Most of the lower score in Spanish accounted for punctuation being tacked on to the end of the previous word in the true tokenization while handled by itself by NLTK. So if there was a period after any word it would get tokenized as "word." The same would be done with numbers. Looking at the languages which had the higher scores, the main issues that persist are the concatenation of symbols and the huge variation in the way people use symbols to express emotions. The NLTK tokenizer also seems to take any amount of consecutive periods to an ellipsis i.e. "....." to "...".

6.2 SpaCy

6.2.1 Quantitative Results

	es	sr	sl	hr
Recall	0.957	0.753	0.680	0.754

Table 7: SpaCy recall scores for the first half of the languages

	da	en	ir	it
Recall	0.581	0.724	0.763	0.796

Table 8: SpaCy recall scores for the second half of the languages

6.2.2 Qualitative results

The spaCy tokenizer[4] performs quite poorly on all languages but Spanish, when compared to NLTK with scores being anywhere around 20% lower across the board bar Spanish. This is majorly due to to handling punctuation and special symbols the same way that they are in the true Spanish tokenization, i.e. "word. word" gets tokenized to ["word.", "word"] rather than ["word", ".", "word"].

The different language modules of SpaCy handle contradictions differently, and seemingly the opposite way of the true tokenization in most cases, splitting them when the true tokenization keeps them as is and vice versa. This is observed in English, Irish, Slovene, Serbian and Italian. In Danish, Croatian it appends the quote mark to one of the words to either side whereas the true tokenization takes it as a single token on its' own.

6.3 Twokenizer

Twokenizer is ”a tokenizer designed for Twitter text in English and some other European languages.”[10]

6.3.1 Quantitative Results

	es	sr	sl	hr
Recall	0.905	0.974	0.968	0.976

Table 9: Twokenizer recall scores for the first half of the languages

	da	en	ir	it
Recall	0.900	0.942	0.971	0.965

Table 10: Twokenizer recall scores for the second half of the languages

6.3.2 Qualitative results

The Twokenizer manages to overcome the challenge of concatenation and emotive symbols well in a lot of cases. It handles both the repeated symbols as well as more exotic combinations such as ”*-*”. Spanish has a low score due to the same punctuation irregularity as before. Interestingly with higher scores a lot more of the inconsistencies and edge cases become apparent. In Slovene, there are a number of low-scoring sentences which have tokens split on nothing. Upon investigation, these are most likely typos. Since the corpus was manually annotated, it would make sense that the person annotating would tokenize to two tokens.
Another discrepancy that becomes more apparent is hyphenation. While a commonly used technique, difficult to handle for the tokenizer due to the true data’s internal inconsistency, even within languages. In some cases hyphens are tokenized as one i.e. ”pro-eu” whereas in others they are not i.e. [”child”, ”-”, ”care”]. While there are rules in the English grammar on how to deal with each case, the twokenizer simply takes any token with a hyphen as a single token.

6.4 MaChAmp

MaChAmp is a a toolkit for easy fine-tuning of contextualized embeddings in multi-task settings. [14] It was trained for tokenization only, with default settings. The model used was mBERT-cased which was shown to work well for tokenization tasks in [12]. mBERT is much less computationally expensive and performs very competitively agaisnt the much more computationally expensive XLM-R which was also evaluated in the paper.

6.4.1 Quantitative Results

	es	sr	sl	hr
Recall	0.928	0.990	0.984	0.984

Table 11: Machamp recall scores for the first half of the languages

	da	en	ir	it
Recall	0.933	0.986	0.988	0.995

Table 12: Machamp recall scores for the second half of the languages

6.4.2 Qualitative results

Achieving very high scores, the trained model does great and allows us to look into some of the hardest cases that a tokenizer can be presented with. It falls short in a number of cases though. Firstly, it has a tendency to over-tokenize, especially in cases of non-anonymized usernames meaning it sometimes tags the next word to the end of the username. The over-tokenization also applies to numbers where it tends to also tag those on to the end of the previous word. Lastly, while it does a great job at finding emoticons, the act of separating multiple units of the same one is still troublesome.
Typos are still a major weakness that are present in every language looked at. Due to the natural sparsity of languages there is an overwhelming chance that the model will not see the same typo in training as it does in the testing/evaluation phase. This is also true for some stylistic choices as for example not putting spaces to make the reader read the words quickly.
Multilingualism shows itself as a more clear challenge as we see the model use tokenizations from other languages. When evaluating croatian, the model clearly uses the tokenization it learnt from English when tokenizing the word ”im”.

7 Discussion

7.1 Summarization of key findings

There are a number of challenges that a tokenizer faces working on multilingual social media data. Firstly, the way that you handle the vast variety of emoticons and other unusual combinations of symbols following each other. While rule-based

tokenizers can be quite good at this, it requires a large amount of logic to be written around it. The fine-tuned model approach works well for this as well.
Secondly, grammar plays a big role. Especially since you are dealing with multiple completely different sets of rules for languages that use the same script. While writing a sophisticated rule-based tokenizer that utilizes a POS tagger to include grammatical rules is feasible for a single language, this approach quickly loses feasibility the more languages you add. Hyphenation and contractions were the main observed culprits, however this is not an exhaustive list.
Typos and non-whitespaced chains of words are very difficult to account for and no tokenizer approach evaluated deals well with them.
Even if using a trained model approach, some rules should be established in the data-cleaning and tweaking stage. Since you are likely working with different datasets, some normalization should take place. In this instance, I would change the punctuation rules of the Spanish data to match the other datasets, anonymize all the usernames into a single format and normalize the link/url formats.
Lastly, due to the "black-box" approach of a fine-tuned model, there is some artifacting which can be troublesome to get rid of or diagnose. In this case the tendency of the model to tag on words to usernames.

7.2 Limitations

A limitation of this project is that all the languages looked at are of eurocentric origin and therefore quite similar in a lot of ways. If expanded, it would be interesting to get some data from languages that differ from the set so far.
Another limitation is the method of analysis. While I believe that the manual qualitative analysis approach is the best way to approach this task due to the nuances and complexity of analysis, even a dataset of this size was already quite overwhelming to go through. A more streamlined approach is definitely needed in case of expansion of the project.

8 Future work

As this project is meant as a basis for experimentation, it is very scalable for future work. Securing more data from more languages is a great start for that. Furthermore, using the knowledge of the common problems presented in this paper to either add to an existing tokenizer or create a new one from scratch is a promising task as well. Additionally, whereas I decided to not touch the data too much for this experiment in order to explore the weaknesses of current methods, there is a lot of merit in making the dataset more robust and consistent.

9 Conclusion

Multilingual auto-tokenization of social media data would save a lot of time and resources as manual annotation is tedious and expensive. Creating a resource to serve as a basis for this research should accelerate the experimentation in this field. Furthermore, looking into already existing solutions is a good first step towards developing something new. The findings of this project provide a starting point for this development.

References

[1] Multi_lang_social github. https://github.itu.dk/ivpe/Multilang_thesis, 2023.

[2] Inaki Alegria, Nora Aranberri, Víctor Fresno, Pablo Gamallo, Lluis Padró, Inaki San Vicente, Jordi Turmo, and Arkaitz Zubiaga. Introducción a la tarea compartida tweet-norm 2013: Normalización léxica de tuits en español. In *Tweet-Norm@ SEPLN*, pages 1–9, 2013.

[3] Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[4] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[5] Jakob Lenardič, Jaka Čibej, Špela Arhar Holdt, Tomaž Erjavec, Darja Fišer, Nikola Ljubešić, Katja Zupan, and Kaja Dobrovoljc. CMC training corpus janes-tag 3.0, 2022. Slovenian language resource repository CLARIN.SI.

[6] Yijia Liu, Yi Zhu, Nathan Schneider, and Noah A. Smith. Tweebank v2. https://github.com/Oneplus/Tweebank, 2013.

[7] Nikola Ljubešić, Tomaž Erjavec, Vuk Batanović, Maja Miličević, and Tanja Samardžić. Serbian twitter training corpus ReLDI-NormTagNER-sr 2.1, 2019. Slovenian language resource repository CLARIN.SI.

[8] Nikola Ljubešić, Tomaž Erjavec, Vuk Batanović, Maja Miličević, and Tanja Samardžić. Croatian twitter training corpus ReLDI-NormTagNER-hr 3.0, 2023. Slovenian language resource repository CLARIN.SI.

[9] Joakim Nivre, Daniel Zeman, Filip Ginter, and Francis Tyers. Universal Dependencies. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, Valencia, Spain, April 2017. Association for Computational Linguistics.

[10] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 380–390, 2013.

[11] Barbara Plank, Kristian Nørgaard Jensen, and Rob van der Goot. Dan+: Danish nested named entities and lexical normalization. *arXiv preprint arXiv:2105.11301*, 2021.

[12] Rob van der Goot. Is tokenization a solved task? 2023.

[13] Rob van der Goot, Alan Ramponi, Arkaitz Zubiaga, Barbara Plank, Benjamin Muller, Iñaki San Vicente Roncal, Nikola Ljubešic, Özlem Çetinoğlu, Rahmad Mahendra, Talha Çolakoglu, et al. Multilexnorm: A shared task on multilingual lexical normalization. In *Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 493–509. Association for Computational Linguistics, 2021.

[14] Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. Massive choice, ample tasks (machamp): A toolkit for multi-task learning in nlp. *arXiv preprint arXiv:2005.14672*, 2020.