Ricardo Lukas Jung
6227492
Empirische Sprachwissenschaft (B.A.)
Phonetik & Digital Humanities
15th Semester
s2458588@stud.uni-frankfurt.de

Bachelor Thesis

Lexicalizing a BERT Tokenizer

Building Open-End MLM for Morpho-Syntactically Similar Languages

Ricardo Lukas Jung

Date of Submission:
January 27, 2023

Text Technology Lab
Prof. Dr. Alexander Mehler
Dr. Zakharia Pourtskhvanidze

Erklärung

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen Quellen oder Hilfsmittel als die in dieser Arbeit angegebenen verwendet habe.

Ort, Datum

Unterschrift

Abstract

This is the abstract: what is this about? what was done? what where the results?

Contents

List of Figures

List of Tables

List of Acronyms

BERT	Bidirectional Encoders from Transformers
CL	Computational Linguistics
GerParCor	German Parliamentary Corpus
LM	Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
MLM	Masked Language Model
NLP	Natural Language Processing
POS	Part of Speech

1 Introduction

2 Overview

3 Methodoloy

in this section the whole methodoloy is covered. what do i use in this thesis, why do i use it and lastly, how? make sure the why covers methodological implications. (vergiss nicht alle pakete als quelle im Anhang)

1 Requirements

A series of tools will help to achieve lexicalized tokenization. They will be explained in this chapter along with their methodological edge.

1.1 Learning Architecture & Tokenizer

Bidirectional Encoders from Transformers (BERT) is a language learning transformer model designed for Natural Language Processing (NLP) tasks (Vaswani et al. 2017). Upon release it achieved higher performance scores compared to previously used Long Short-Term Memory (LSTM) models (Devlin et al. 2018). Two main model characteristics can be observed for BERT. Firstly, it is the first Language Model (LM) to implement simultaneous attention heads, allowing for bidirectional reading. The methodological implication of reading to the left and right of a token is to include more information about the language in single embeddings. Secondly, BERT introduced the (at the time novel) Masked Language Model (MLM) method for training. The method involves masking a specified amount (default 15%) of random tokens in the input sequence. Masked tokens are guessed by the model which can then update its weights according to success or failure.

The NLP community has since developed BERT and adapted it to the needs of contemporary NLP problems (roberta, germanbert, mbert CITATION). Its wide support, comparability and versatility make BERT the model of choice for this thesis. Another notable feature in BERT is the implementation of the WordPiece tokenizer module (QUELLE?). Default BERT WordPiece tokenization is predominantly heuristic by combining strings based on a precalculated score. A variety of pre-trained tokenizers are available, although they come with a caveat. Once a tokenizer is trained on a dataset it is specific to that dataset. This means the application of a tokenizer on another dataset may result in out-of-vocabulary issues and different token/subtoken distributions.

Particularly relevant to this thesis is the option to train an own tokenizer from the base module. Usually, WordPiece generates its own set of subtokens called *vocabulary*. Tokens are then WORDPIECE ALGORITHMUS ERKLÄREN By providing an algorithmically generated vocabulary to WordPiece and then training it on a new dataset the tokenization behavior is changed.

1.2 Data

explain the data that is used

1.3 Benchmark

explain olmpics

2 Implementation

Tatsächliche Anwendung der Methoden auf die Daten

2.1 Tokenizer

ESSENTIALLY DERIVING SENSIBLE SUBTOKENS TO REPRESENT LEXEMES

Generating a custom pre-training vocabulary

The Wordmap algorithm as shown in Algorithm 1 is the first step to extracting morphemes from a token. Its purpose is to compare two strings and store their intersections in a map of boolean values.

Wordmap requires two **inputs** $verbs$ and $target$. The resulting wordmap will be generated for $target$, while $verbs$ serves as comparison. $verbs$ is a set of tokens pertaining to the same Part of Speech (POS) category. Note that $verbs$ should only contain those POS-tagged tokens that are expected to carry lexical information (e.g. verbs, adjectives, etc.). The set is previously extracted from the corpus by POS-tagging. Optionally, the set can be augmented by manually adding POS matching tokens from external sources. The 2-tuple $pair$ are the strings to be compared. It is passed on to (1) SHORTER, a function returning the shorter of both strings (2) LONGER, expectedly returning the longer of both strings (3) MATCH_CASE, a function to determine the behavior of the algorithm later on. As two strings are compared MATCH_CASE captures three cases: $pair$ matches in the first, last or both positions. Finally len denotes the length of the longest string and δ difference in length between s and l. δ functions as an offset for index-based comparisons in WORDMAP.

WORDMAP is the function responsible for generating the wordmaps.

Once every v has been compared to $target$, $maps$ boolean counts of characters occurring in their respective positions in $target$. Every map is cleaned with a regular expression to reduce noice caused by natural character occurrence. Continuous concatenations of leading or trailing matches stay, while every match with the word enclosed by 0 will be replaced the latter. As an example, $wordmap = 11101100101$ contains three matches on the inside which will result in 11100000001 as final output. In the penultimate stage of mapping $target$, all maps are summed up to receive the number of absolute positional occurrences of every character in $target$. This positional mapping allows for detecting relevant segments in a token based on a threshold. Characters in range of the predefined threshold are selected for the mapping of a target token. Functional morphemes (morphemes that are carriers of grammatical features) are typically much more frequent than their lexical counterparts. Consequently,

Algorithm 1 Wordmap generation

```
Input: $verbs = \{v : v \in C \land v_{POS}\}, target$        ▷ Set of single-POS lexemic tokens
Output: $maps = (map_1, \ldots, map_{|verbs|})$
 1: $pair = (target, v)$
 2: $s = \textsc{shorter}(pair)$
 3: $l = \textsc{longer}(pair)$
 4: $case = \textsc{match\_ends}(pair)$        ▷ Returns if strings match in the last or first position
 5: $len = \textsc{len}(l)$
 6: $\delta = \Delta(len - \textsc{len}(s))$
 7:
 8: function WORDMAP(w1, w2, d=0)
 9:     $f_1 : c1, c2 \mapsto c1 == c2$
10:     $f_2 : \left(\sum_{i=0+d}^{|w1|} w1[i], w2[i] \mapsto f_1(w1[i], w2[i])\right)$
11:     return $f_2$
12: end function
13:
14: if case: any match then
15:     if $\delta$ then
16:         if case: left match then
17:             $\textsc{wordmap}(l, s)$
18:             Pad map from right side with 0s to match $len$
19:         end if
20:         if case: right match then
21:             $\textsc{wordmap}(l, s, \delta)$
22:             Pad map from left side with 0s to match $len$
23:         end if
24:     else
25:         $\textsc{wordmap}(l, s)$
26:     end if
27: end if
```

String	Wordmap	Case	Padding
`verarbeiten`	111000000	left match	Yes
	001000011	right match	Yes
`variiert`	101001000	left match	Yes
`vormachen`	101000111	left match	No
	101000111	right match	No
`anstrebtest`		no match	
(...)			

Table 3.1: Example wordmaps for $target$ = `verstehen`

lexical morphemes in the family of inflectional languages are - by definition - modified by functional morphemes, they occur much less frequently. In this case, the activation function for a concatenation of same boolean values to be selected as segment is the normalizing z-score function defined as: $z_i = \frac{x_i - \overline{x}}{S}$, where z is the z-score, $\overline{x}$ and S are the sample mean and the sample standard deviation. Interestingly, thresholds

Tokenizer Training

How did I train the tokenizer, how did it go? Which problems arose? What went well? What happened?

2.2 Masked Language Model

Model implementation and parameters, runtimes?

2.3 oLMpics Benchmark

tweak des tokenizers: segmentation ist eine frage der interpretation. The Ultimately, segmentation is a matter of interpretation. As mentioned in 1.1, the default WordPiece Tokenizer lacks A linguistically informed

The field of NLP (Glück and Rödel 2016) has been expanded ever since the emergence of the language models. Natural language processing is understood as the

4 Results

5 Discussion

6 Conclusion

7 Testchapter

1 Citing

Abrami et al. 2022

2 Quoting

> "This is a quote by textquote" (DeepL 2021) "This is a quote by enquote"

3 Referencing

Short reference 1.1
Long reference subsection 1.1
monofont for code or string `monofont`

Bibliography

Abrami, Giuseppe, Mevlüt Bagci, Leon Hammerla, and Alexander Mehler (June 2022). "German Parliamentary Corpus (GerParCor)". In: *Proceedings of the Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, pp. 1900–1906. URL: `https://aclanthology.org/2022.lrec-1.202`.

DeepL (Nov. 2021). *How does deepl work?* URL: `https://www.deepl.com/en/blog/how-does-deepl-work`. Last accessed: 28.12.2022.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. DOI: `10.48550/ARXIV.1810.04805`. URL: `https://arxiv.org/abs/1810.04805`.

Glück, Helmut and Michael Rödel, eds. (2016). *Metzler Lexikon Sprache*. ger. 5th ed. Springer eBook Collection. Stuttgart: J.B. Metzler, pp. 141–142. ISBN: 978-3-476-05486-9. DOI: `10.1007/978-3-476-05486-9`. URL: `http://dx.doi.org/10.1007/978-3-476-05486-9`.

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. DOI: `10.48550/ARXIV.1706.03762`. URL: `https://arxiv.org/abs/1706.03762`.