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:
February 6, 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

This thesis showcases the use of specific intervention in tokenization subsystems of machine learning. The intent of this thesis is to inject linguistic bias into the machine learning framework of BERT to sharpen the analytical capacities of a masked language model. In this chapter the background, intentions and scope of the thesis are covered.

1 Motivation

WHY IS THIS SUBJECT RELEVANT There is an ongoing urge in the Computational Linguistics (CL) community to understand natural language. Research in the past decades shows use of frequentist and statistical methods (such as gpt and bert ZITATION) to their advantage, leading to the emergence of the first machine learning (ML) models. It became apparent that these ML models are the best currently available approach to an automated understanding of natural language. The structural parallels of machine learning to human learning have often been drawn (ZITATION)) to demonstrate how similar and more importantly: how different both can be. A powerful feature of Machine Learning (ML) (as opposed to human learning) is the possibility of actively controlling the the learning parameters in a supervised environment. To test the efficiency of ML parameters a variety of tasks (ZITATION) are designed and applied. A trained model will yield performance scores based on the quality of its training, much like humans on language tests. But the automated modeling of language is not the first instance of language modelling in a broader sense. Traditional linguistics (DEFINITION has produced fundamental research the prior to the discovery of ML architectures and their implementation. While generic ML frameworks seem appealing in the presumption that they require less work to reach somewhat satisfactoy results, their performance is still incomplete. The integration of aforementioned traditional linguistic knowledge into learning processes for machine learning is the underlying motivation of this thesis.

flota FLOTA Language learners usually build up a lexicon consisting of lexemes which they will have to analyze accurately in order to be productive in that target language. A ML model relies on a tokenizer to create such a vocabulary (**ZITATION**). It is programmed to segment tokens into subwords (if possible) and provide a vocabulary comprising all the components needed to analyze a given string. Ideally those subwords will be part of the functional vocabulary in the target language, which would make them a morphemes **ERKLÄRUNG**. A morpheme is canonically defined as the smallest unit carrying meaning in a language **morpheme**. The problem of captchuring lexical morphemes at the tokenization level is a problem of morphological analysis. A standard approach implemented in the BERT architecture is WordPiece tokenization, which takes pairs of ngrams and calculates a score based on the overall frequency in the text as well as cooccurrences. This was idea pioneered by Schuster and Nakajima (2012) to deal with the problem of near infinite vocabularies and was later adopted in a google research project designing the final WordPiece algorithm

WORDPIECEGOOGLE. WordPiece is designed to find the most effective pairs of subwords in a corpus to optimize vocabulary size. Subwords however are not necessarily a productive component of inventory that is used for inflectional morphology. In fact, the morphemes of a language and its generated tokenizer vocabulary coincide.

Following the guiding principle that **input quality is ouput quality** not only in language learning, the morpheme vocabulary is identified as the point of leverage in the upcoming section. Note: explain why i use tokens and words, they are interchangeable right? holistic, need less attention to produce satisfactory NOT JUST TO PUSH F, BUT TO FIND A VIABLE METHOD OF MORPHEMIC TOKENIZATION

2 Hypotheses

The following research questions will be formulated for testing:

HYP1: Adjustments to tokenization have significant impact the performance of a language model in different tasks.

How to achieve this hypothesis?

HYP2: Providing lexical information to a tokenizer increases benchmark accuracy on MLM tasks.

How to achieve this hypothesis?

3 Scope and Structure

The following chapters are sorted into three parts. To outline the research domain, a brief summary of the current state of morphological language modeling is given. Next, german is described paying special attention to its morphological complexity and peer languages. This serves as preface to the methodology, connecting characteristically matching languages to form a pool of possible target languages.

As main part of this thesis, the methodology is layed out. It is sectioned into a theoretical part which focuses on what implements are used and the value they hold towards lexicalizing a tokenizer

What is covered and what not? What is the shape of this thesis and what order does it have?

2 Overview

define morpheme vs subword define morphological tokenization

Morphological tokenization can be understood as the process of identifying segments in text that are a productive in a given language, carrying meaning and hence also fitting the definition of a morpheme. describe the most recent findings on morphologically pretrained models in machine learning literature findings on POS effect on ML

1 Related Works

In the past, many efforts towards morphological tokenization have been made. This thesis was mainly inspired by the FLOTA

Well known attempts like morfessor **morfessor** have been outperformed by Sequence based models that also use linguistic morphology. Notably, top-down generation of subword vocabularies has shown promising results for tokenization in fusional languages **subwordvsmorfessor**. This aligns with the notion that standard BPE or WordPiece tokenization effectivity suffers from complex morphology causing a big vocabulary. The overall comparison **subwordvsmorfessor** shows an increase in performance for languages of similar morphological complexity. It is interesting to see that this form of tokenization performs less well for English, compared to languages in morphology than its agglutinative fusional peers, e.g. Italian, Latin, Spanish, Russian flota and wordpiece tokenizer themselves are crude shot

2 Target Languages

This section identifies target languages that share common morphological features with German. It is assumed that languages of the same morphological type will behave similarly when analyzed morphologically. German was selected to serve as example language within the family of fusional languages. The aim is not to propose yet another case study of German, but to introduce German as a surrogate to further the scope of application on other languages of similar morphological complexity.

Describe german (ISO639-3: deu) and its morphological state. Compare to other languages with interlinear glossing.

German (ISO639-3: `deu`) is a west-germanic language and the official language of Germany, Austria, Switzerland, Liechtenstein and Luxemburg (Glück and Rödel 2016). It is an inflectional synthetic language with approximately 130 million speakers[1]. German is largely researched and is still paid much attention to in the domain of (computational) linguistics.

[1] https://de.statista.com/statistik/daten/studie/1119851/umfrage/deutschsprachige-menschen-weltweit/ Last accessed: 09.01.23

(1) My s Marko poexa-l-i avtobus-om v Peredelkino
1PL COM Marko go-PST-PL bus-INS ALL Peredelkino
'Marko and I went to Perdelkino by bus.'

2.1 Pooling similar languages

On one side, linguistic typology has come up with many useful classifications for languages. On the other, in the pursuit of reconstructing languages Indo-European studies have established a widely accepted phylogenetic model of the diachronic dependecy of Indo-European languages. Both disciplines contribute to language classicifications that are used in this subsection.

Morphological complexity is a term to describe how languages use paradigms to connect grammatical information with lexemic information (Baerman, Brown, and Corbett 2017). Mind that morphological complexity is a nominal category to describe gradients of function-to-morpheme correspondence (**QUELLE**, not a qualitative assessment. The common denominators that make languages morphologically complex are their morphological features. Those languages that use affixation, fusion, composition and derivation (among others) are all fit candidates compared to german.

A summary of morphological typology is provided in 2017, pp. 78–93).

Thus, German will be the exemplary target language for the experimental setup.

Typological findings on Indo-European languages . and by means of composition and derivation.

Describe what morphological complexity is.

Bearman, Brown and Corbett

Describe what similar languages exist (typological vs topological)

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 wiktionary verbs hanta-crawled verbs of gerparcor oscar de gerparcor bundestag subset

1.3 Benchmark

explain olmpics

2 Implementation

Tatsächliche Anwendung der Methoden auf die Daten

2.1 Tokenizer

ESSENTIALLY DERIVING SENSIBLE SUBTOKENS TO REPRESENT LEXEMES The methods of tokenization is explained in the following. This is the main part of the thesis

Throughout this section, the words $target$ and $token$ are used to describe a word that is analyzed. One denotes the argument of the wordmapping and segmenter function ($target$), the other describes a word occurring in the corpus ($token$).

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. Any map generated from this also has the same length as $target$. $verbs$ 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 WORDMAP.

WORDMAP is a naive mapping function 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

Algorithm 1 Wordmap generation

Input: $verbs = \{v : vv_{POS}\}, target$ ▷ Set of single-POS lexemic tokens

Output: $maps = (map_1, \ldots, map_{|verbs|})$

function WORDMAP(w1, w2, d=0)
    $f_1 : c1, c2 \mapsto c1 == c2$
    $f_2 : \left(\sum_{i=0+d}^{|w1|} w1[i], w2[i] \mapsto f_1(w1[i], w2[i]) \right)$
    return f_2
end function

for $v \in verbs$ **do**
    $pair = (target, v)$
    $s = \text{SHORTER}(pair)$
    $l = \text{LONGER}(pair)$
    $case = \text{MATCH\_ENDS}(pair)$ ▷ Returns if strings match in the last or first position
    $len = \text{LEN}(l)$
    $\delta = \Delta(len, \text{LEN}(s))$

    if *case: any match* **then**
        if δ **then**
            if *case: left match* **then**
                WORDMAP(l, s)
                Pad map from right side with 0s to match len
            end if
            if *case: right match* **then**
                WORDMAP(l, s, δ)
                Pad map from left side with 0s to match len
            end if
        else
            WORDMAP(l, s)
        end if
    end if
end for

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`

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, 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.

BERT Tokenizer Modification

What is the model tokenizer mod Which problems arose? What went well? What happened?

2.2 Masked Language Model

Model implementation and parameters, runtimes?

2.3 oLMpics Benchmark

Algorithm 2 recursively computes every possible segmentation for a string $target$ from a given vocabulary pos_vocab from left to right. The vocabulary contains all the segments that were identified in the previous step by Algorithm 1. Every segmentation has to be complete so that its segments corresponds to non overlapping substrings of $target$. For every $target$ a subvocabulary $morpheme$ is defined, containing all strings that are in the vocabulary of **POS! (POS!)** members. This task is a weighted coverage problem in the NP-hard domain. Unary morphemes to the left are excluded from the pre-selected subvocabulary to drastically reduce the number of possible permutations , as they can be embedded in n-ary tokens as well. The recursion can be seen as n-ary a trees containing every permutation of the set $morphemes$ where the sum of branches all satisfy $target$.

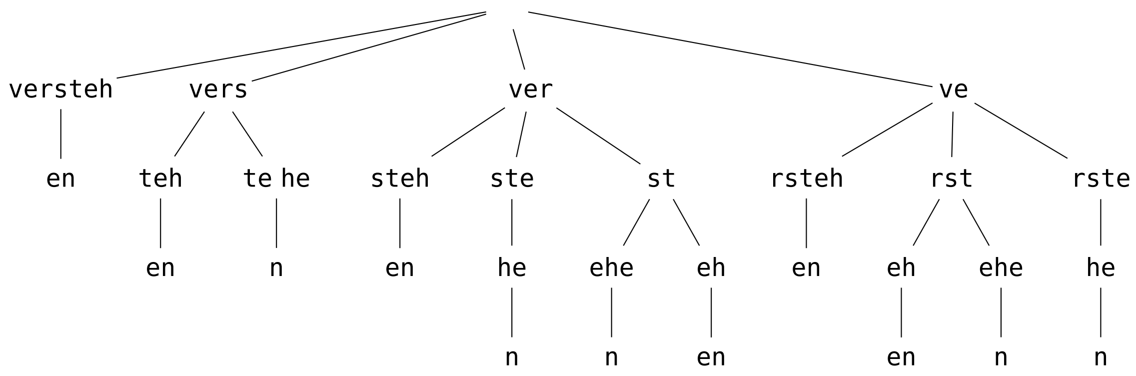
Algorithm 2 Target Segmentation

Input: $target, pos\_vocab$ ▷ short comment

Output: $\{(tuples\ of\ subwords)\} \approx \{t \in \mathcal{P}(s \in pos\_vocab : s \in target) : t \equiv target\}$

$segmentations = ()$

function SEGMENTER(token, stop, start=0, segments)
    if $start = stop$ **then**
        Add $segments$ to $segmentations$
    else
        $morphemes = (m \in pos\_vocab : target.\text{STARTSWITH}(m) \wedge |m| > 1)$
        for $m \in morphemes$ **do**
            $start \mathrel{+}= \text{LEN}(m)$
            Add m to $segments$
            $rest = target[\text{LEN}(m) :]$
            if $\text{LEN}(rest) == 1$ **then**
                $start = stop$
                Add $rest$ to $segments$
                Add $segments$ to $segmentations$
                Decrement $start$, crop $segments$
            else
                $\text{SEGMENTER}(target = rest, stop, start = start, segments)$
                Decrement $start$, crop $segments$
            end if
        end for
    end if
end function
$\text{MAXIMIZE\_SEGMENTS}(segmentations)$



First, all morphemes that $targets$ starts with are stored to form the first nodes of the permutation trees. Each time a morpheme is selected the index $start$ is incremented by the length of the morpheme to indicate when the string has been completely segmented. The new recursion is called with the updated index $start$ and $target$ sliced by the legnth of the morpheme contained in the parent node. Incomplete segmentations that miss exactly one character to the right are accepted with the added missing string. If the vocabulary cannot satisfy a segmentation by missing the necessary strings segmentation is omitted and the original input token is returned as such.

Then, of all $segmentations$ a single segmentation is selected by a maximization function MAXIMIZE_SEGMENTS calculating weights for every segment. The maximization function for one segment is the defined as:

$$\arg\max_{s \in S} f(x) := \{\sum_{i=1}^{|s|} \sqrt[m]{\frac{s_i}{t} \div |s|}\} \tag{3.1}$$

Where s is a segmentation tuple, $|s|$ is the length of the tuple (read: number of segments) and s_i indicates the segment of the tuple at position i. For every segment (vertex) in a segmentation tuple (branch) the segment's coverage length is divided by the token length t to get the coverage of the morpheme

tweak des tokenizers: segmentation ist eine frage der interpretation. The Ultimately, segmentation is a matter of interpretation. If there are several possible interpretations by wich to segment a word this tokenization method relies on the assumption that the comparison with POSmembers displays the probable shape of segmentation for a token. 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.

Aikhenvald, Alexandra and R Dixon (2017). *The Cambridge Handbook of Linguistic Typology*. Cambridge Handbooks in Language and Linguistics. Cambridge: Cambridge University Press. DOI: 10.1017/9781316135716.

Baerman, Matthew, Dunstan Brown, and Greville G. Corbett, eds. (2017). *Morphological Complexity*. Cambridge studies in Linguistics 153. Cambridge University Press. ISBN: 1107120640. DOI: 10.1017/9781316343074. URL: www.cambridge.org/9781107120648.

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.

Schuster, Mike and Kaisuke Nakajima (2012). "Japanese and Korean voice search". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152. DOI: 10.1109/ICASSP.2012.6289079.

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