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 13, 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

**bbgc** bert-base-german-cased

**BERT** Bidirectional Encoders from Transformers

**BPE** Byte Pair Encoding

**BWE** Bert WordPiece

**CL** Computational Linguistics

**GerParCor** German Parliamentary Corpus

**HanTa** Hanover Tagger

**LM** Language Model

**LSTM** Long Short-Term Memory

**ML** Machine Learning

**MLM** Masked Language Model

**NLP** Natural Language Processing

**POS** Part of Speech

**TTLab** Text Technology Lab

**WM** Wordmap

arabic

# 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 Bidirectional Encoders from Transformers (BERT) to sharpen the analytical capacities of a masked language model. In this chapter the background, intentions and scope of the thesis are covered.

## 1.1 Motivation

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 (Radford et al. 2018) and bert Vaswani et al. 2017) to their advantage, leading to the emergence of the powerful, problem solving 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 (Jianlong Zhou; Kühl et al. 2020) to demonstrate how similar and also how different both can be. An essential feature of ML (as opposed to human learning) is the possibility of actively controlling the learning parameters in a supervised environment. To test the efficiency of ML a variety of tasks (e.g. classification, clustering, detection) are designed and applied. A trained model will yield performance scores based on the quality of its training, much like humans on language tests. Yet, automated modeling of language is not the first instance of language modelling in a broader sense. Traditional linguistics has produced fundamental research 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.

While the research on neurophysical intake and representation of language is not entirely conclusive (Delogu, Brouwer, and Crocker (2019, pp. 1–3); Kimppa et al. (2019)), it is assumed that language learners usually build up a lexicon consisting of lexemes (Brennan 2022, pp. 82 – 100) which they will have to analyze accurately in order to be proficient in that target language. A ML model relies on a tokenizer to create such a vocabularies (**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 Colman 2009. 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 Wu et al. 2016. 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.

Hofmann, Schuetze, and Pierrehumbert 2022

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

## 1.2 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?

## 1.3 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 for a new tokenization method is layed out. It is sectioned into a theoretical part which focuses on the architecture of its components and the value they hold towards lexicalizing a tokenizer. The implementation of each part in the tokenization process is explained next.

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

# 2 Overview

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

## 2.1 Related Works

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

Earlier generalized attempts like morfessor (Creutz and Lagus 2002) have been outperformed by Sequence based models that also use linguistic morphology  Peters and Martins 2022 . Notably, top-down generation of subword vocabularies has shown promising results for tokenization in fusional languages. This aligns with the notion that standard BPE (Sennrich, Haddow, and Birch 2016) or WordPiece  Wu et al. 2016 tokenization effectivity suffers from complex morphology causing a big vocabulary. The overall comparison  Peters and Martins 2022, p. 134 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, a language that has seen a decline in morphology. Much better benchmarks are reached applying its agglutinative fusional peers, e.g. Italian, Latin, Spanish, Russian Toraman et al. find that the vocabulary size plays a special role in morphological tokenization and even define a ratio vocabulary size ratio between 20 to 40% to the number of model parameters depending on the type of tokenizer ( Toraman et al. 2022, pp. 11–12). Since tokenization and vocabulary are obviously interdependent, the vast amount of typological variety seen in languages raises the question: is there a right way of tokenizing? This issue is addressed by Rust et al., where a mid-scale investigation was done to see whether different languages actually need more specific tokenizers compared to generalized tokenizers. They report an improvement of model accuracy and F-score across all tasks and languages (Rust et al. 2020). While this sketches a commission for Natural Language Processing (NLP) to always consider choosing a method tailored for single languages, the answer to the problem of performance versus maintenance in models might not be as elaborate as treating every language singularly. Every language is undoubtedly unique, but that does not rule out simplification by means of further classifying and grouping target languages. In an effort to explain the provenience and relatedness of languages many tools in the domain of typology, NLP and indo-european studies have been constructed.

Whether it be identification of morphological features (Comrie 1989, pp. 42–56), complexity measures (Çöltekin and Rama 2022) or connection through reconstruction (Bouckaert et al. 2012), the different linguistic disciplines suggest observable regularities by which to morphologically group languages. Leveraging the relatedness of languages in NLP is not a new idea in tokenization or Part of Speech (POS) -tagging, but is seeing mixed results up to this day, even with augmentation methods (Aepli and Sennrich 2021). The options seem to branch out quickly, but the mechanism of clearly separating lexemic and functional information seems inherent to most languages. The way they differ is in they combine grammatical functions in morphemes (fusion) and bind them to lexemic morphemes (synthesis). This may be why approaches with stemming, lemmatization or other morphological analyses are very relevant to building good tokenizers for all languages on the isolating to synthetic spectrum (Schwartz et al. 2020, pp. 51–53).

## 2.2 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.

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.

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

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

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 and measure of morphematic agreement, 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).

Due to the scope of this thesis, German will be the exemplary target language for the experimental setup. Its morphological complexity can be compared to other related or non-related languages as shown in Table 1. There still is no universally accepted measure the complexity of a language due to , but groupings exist on different parameters:

| **Language** | **Similarity** | **ISO 639–3** |
|---|---|---|
| Norwegian | Closely Related | isl |
| Danish | Closely Related | nor |
| Dutch | Closely Related | nld |
| English | Closely Related | eng |
| Icelandic | Closely Related | isl |
| Romanian | Morphology | ron |
| Spanish | Morphology | spa |
| Finnish | Morphology | fin |
| Italian | Morphology | ita |
| Hungarian | Morphology | hun |

Table 1: Listing of languages similar to German given the type of similarity based off Lehman (2022) and Ehret et al. (2021). ISO identifiers provided at WALS[2].

There have been interpolations between human judgements and statistical measures (on similarity of languages) which can be taken into consideration (Bentz et al. 2016). The point to be taken is that while there is no definite proof of concept for tokenizations being effective when connecting target languages through morphological parameters, there is a strong suggestion in data and intuition of researchers that tokenization for morphologically similar languages should profit from these similarities.

With an arguable exception to English, the languages in Table 1 treat their lexemes with similar morphological processes. The upcoming interlinear glossings (as per Leipzig Glossing

Rules[3]) provide examples for inflectional morphology in verbs within this group of languages. To outline word formation processes, a glossing from Nikanne (2017, p. 71) is considered:

(1) Tytöt istu-i-vat tuolilla
girls sit-PST-3PL chair.ADE
'The girls sat on the chair.'

This Finnish sentence is a textbook example of agglutinative inflectional morphology. The verb {istu} is inflected by suffixing two morphemes marking the past tense {i} and the third person plural {vat} (curled brackets denote morpheme boundaries). In this case every morpheme expresses one grammatical function, apart from {istu} which contains the lexical information for the verb "to sit ". The functional morphemes in example (1) follow the word to be inflected. With many more functional morphemes present in Finnish, Nikanne reports that there is an order in which inflectional morphemes usually appear. In consequence, analyzing Finnish verbs results in different but reoccuring patterns depending on the degree of inflection. The lexemic morpheme {istu} can be modified by {i} alone to just express past tense and still be productive. Following up on the idea of agglutination, Hungarian applies an slightly different strategy to achieve inflection:

(2) Tegnap meg-hallgattunk egy lányt.
yesterday PRF-listen.PST.1PL a girl.ACC
'Yesterday we interviewed a girl.'

Hungarian is also classifed as an agglutinative language for its frequent use of affixes. It is additionally known to combine several grammatical functions into one morphene as can be seen in example (2) as given by Kiss and Hegedus (2021, p. 262), making it a hybrid of agglutinative and fusional. The analysis of the verb in (2) does not allow canonical segmentation to the stem although there is an underlying form {hall;*} meaning "hear ". Instead, it exists in inflection paradigms like the given example {hallgatunk} combinding the grammatical categories past tense and first person plural. The morphemes addressed so far where either suffixed or not segmentable. As lexical part {hallgatunk} receives a prefix {meg} expressing perfect tense. This use of morphemes can also be seen in germanic or romance languages, like Italian (adapted from Iacobini and Masini (2005, p. 163)):

[3]https://www.eva.mpg.de/lingua/pdf/Glossing-Rules.pdf

(3) far=se=la sotto
do-REFL.PRT-PRON.PRT under
'To quake in one's boots''

Here the {se} and {la} carry two functions each and are suffixed to {far}, showing that there are morphological types in between agglutinating and fusioning. Arguably, {se} being a clitic pronoun that will appear in different positions acting as indirect object, but never independent of the verb.

After a partial look on the classified languages two important empirical descriptive caveats remain: there are exceptions to almost every regularity in languages. No language is entirely consistent in following a morphosyntactical paradigm, meaning no language is entirely fusional or agglutinative (same applies to the synthetic and isoling spectrum). Judging from the word shapes in the data and literature, the way languages modify their stems or lexemic morphemes is largely based on affixation. In a tokenizer acknowledging lexemic parts of words, the knowledge of word formation in the target language should be conveyed.

# 3 Methodology

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) The

## 3.1 Requirements

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

### 3.1.1 Pipeline Structure & Transformers Library

BERT is a language learning transformer model designed for 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.

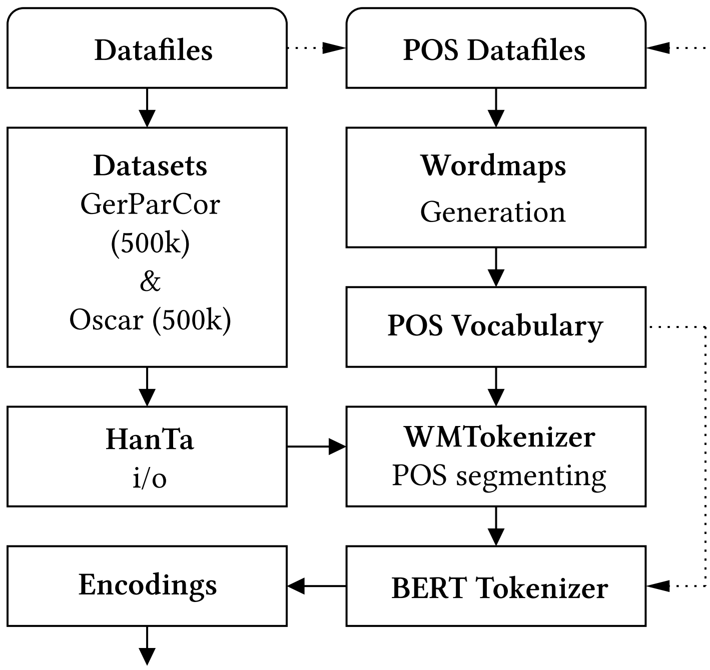


Figure 1: Schematic integration of Wordmap segmentation into the BERT architecture. Continuous lines show non-selective channels, while dotted lines only pass data selectively

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*, which is used by the standard tokenizer to generate unique input numerical IDs. These identifiers correspond to tokens or parts of tokens found in the original dataset that the model was trained on. Wordmap partly takes over the tokenization depending on the the type of input it receives, the exact process is given in subsection 3.2.1. Once a string is tokenized it gets passed on to the transformer model for contextualized embedding. Schematic Figure 1 shows the processing of data up until token encoding.

In Figure 1, Datafiles contains the raw unprocessed data coming from corpora. The input is provided in txt-files holding sentences line by line. Two datasets are generated from this input, each amounting to 500000 sentences per corpus for fine-tuning. POS Datafiles ideally contains only words that are identified as the same POS category. To increase the resulting POS vocabulary, POS Datafiles can be augmented by two channels: input can come from the corpus itself, but would need to be tagged first. Otherwise a predefined list of words can be used as an external and generic source for Wordmap generation. In POS Vocabulary, naive adjustments like removing outliers and interferences are performed on the set of subwords yielded by Wordmap generation. It is important to note that the POS vocabulary is separate from the the tokenizer vocabulary. This serves the specific purpose of keeping the vocabulary on which the model is trained clean from unwanted or unused subwords. Machine learning practice generally points to the trade-off between vocabulary size and model performance, hence the addition of POS Vocabulary to the BERT Tokenizer vocabulary is used only when necessary. Datasets are then passed through Hanover Tagger (HanTa) to provide WMTokenizer with flagged data to tokenize only selected tokens. Lastly BERT Tokenizer returns the canonical encodings known from language modeling.

### 3.1.2 Data

Two corpora where selected for fine-tuning: German Parliamentary Corpus (GerParCor) and oscar (OSCAR 2022). FLOTA was trained on 12000 samples per category Hofmann, Schuetze, and Pierrehumbert 2022, which is why for this fine-tuning a sample size of 500000 seems sufficient. GerParCor is a "GerParCor genre-specific corpus of (predominantly historical) German-language parliamentary protocols from three centuries and four countries, including state and federal level data." (Abrami et al. 2022, p. 1). Of all subcorpora included in GerParCor, one specifically texts and transcripts of the german parliament. All 500k samples in the GerParCor dataset used for this thesis is found in the `Bundestag` subcorpus of GerParCor. Oscar offers other subcorpora partly hosted on huggingface. In this case, a web-crawled german corpus called `unshuffled_deduplicated_de` was chosen, as introduced by its curators Ortiz Su'arez, Sagot, and Romary (2019); (2020).

The POS vocabulary comes from Wiktionary 2022 hanta-crawled verbs of gerparcor Wartena 2019

### 3.1.3 Benchmark

The benchmark used on the model is oLMpics (Talmor et al. 2020)

## 3.2 Implementation

Tatsächliche Anwendung der Methoden auf die Daten

### 3.2.1 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 containing two algorithms for vocabulary generation and token segmentation. Once the two algorithms have been explained, the last section of this chapter features the

Languages, as consistent form of communication, always 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

Embedding those subwords which take part in inflectional processes essentially means deriving sensible subwords to represent actual morphemes.

The vocabulary generated for this chapter is a list of inflected and non-inflected verbs (as to state the example) provided by two sources: (1) the german verb wiktionary and (2) a crawl of the GerParCor corpus with the HanTa tagger Wartena 2019. Initial experiments where done only with the wiktionary verb list since it provided sanitized input for the algorithm in development. As soon as the verbs where extracted from GerParCor via HanTa

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 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 $\delta$ difference in length between $s$ and $l$. $\delta$ 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 $target$, $maps$ stores boolean counts of characters occurring in their respective positions in $target$. Every map is cleaned with a regular expression to reduce noise caused by natural character occurrence (some characters like <n>,

**Algorithm 1** Wordmap generation

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

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

```
function WORDMAP(w1, w2, d=0)
    f_1 : c1, c2 ↦ c1 == c2
    f_2 : ( Σ_{i=0+d}^{|w1|} w1[i], w2[i] ↦ f_1(w1[i], w2[i]) )
    return f_2
end function

for v ∈ verbs do
    pair = (target, v)
    s = SHORTER(pair)
    l = LONGER(pair)
    case = MATCH_ENDS(pair)        ▷ Returns if strings match in the last or first position
    len = LEN(l)
    δ = Δ(len, 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
```

Table 2: Example wordmaps for $target$ = `verstehen`

| 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 | |
| (...) | | | |

<s> will be more frequent than others). Continuous concatenations of leading or trailing matches stay, while every match enclosed by 0 will be replaced the 0. 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, 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. Any segment below a z-score of zero is detected. For every verb that is mapped, the detected affixes are added to a vocabulary of functional morphemes. The rest of the string is saved into the vocabulary of lexemic segments, assuming that it still contains the lexical information in the verb.

Both vocabularies are then cleared of outliers in different ways. The functional vocabulary drops every string longer than 2.5 times the standard deviation of its own population. Since more variance in length is expected in the lexemic vocabulary, the outlier function is limited to drop everything above the length of 2 times the mean absolute deviation as a staple method for more volatile datasets. The upper segment of n-length strings in the lexemic vocabulary has thus been covered, but the low segment (short strings) remains untouched. To solve this problem, probably one of the more reaching but important measures is taken: removing any string of the lexemic vocabulary that is smaller than the mean length of strings in the functional vocabulary. Removing this interference in vocabularies is subject to the notion that functional morphemes will be shorter than lexemic morphemes. After minimizing outliers and interference, both vocabularies are joined again to form the segmenters vocabulary.

### BERT Tokenizer Modification

**Algorithm 2** Target Segmentation

**Input:** $target, pos\_vocab$ ▷ Vocabulary consisting of verbs only
**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**
    $morphemes = (m \in pos\_vocab : target.\textsc{startswith}(m) \wedge |m| > 1)$
    **for** $m \in morphemes$ **do**
      $start \mathrel{+}= \textsc{len}(m)$
      Add $m$ to $segments$
      $rest = target[\textsc{len}(m) :]$
      **if** $\textsc{len}(rest) == 1$ **then**
        $start = stop$
        Add $rest$ to $segments$
        Add $segments$ to $segmentations$
        Decrement $start$, crop $segments$
      **else**
        $\textsc{segmenter}(target = rest, stop, start = start, segments)$
        Decrement $start$, crop $segments$
      **end if**
    **end for**
  **end if**
**end function**
$\textsc{maximize\_segments}(segmentations)$

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.

| Call: | SEGMENTER("verstehen") |
|---|---|
| Subvocab: | {versteh vers ver ve en teh steh ste st rsteh rst rste ehe he n} |

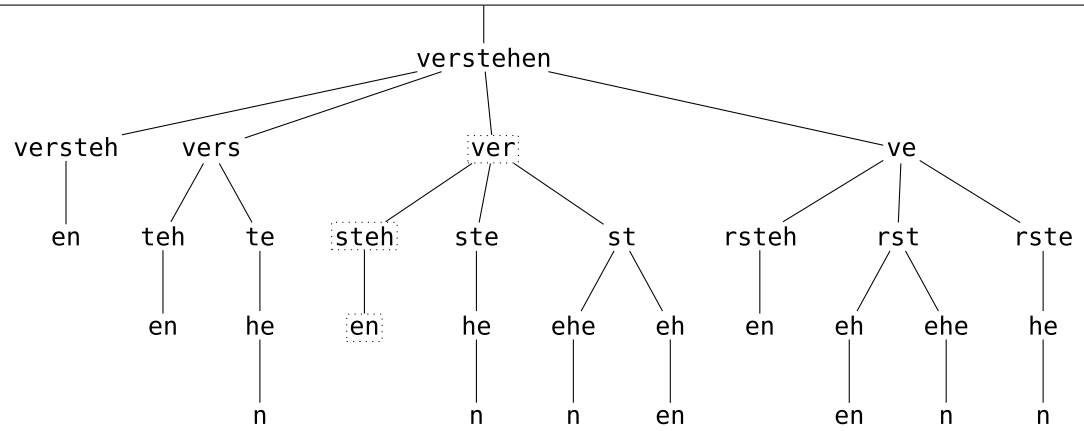


Figure 2: Permutations SEGMENTER generates from target i.e. `verstehen`. Dotted segments mark the segmentation chosen later by the maximization function during tokenization.

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 $target$.

As shown in Figure 2, all morphemes that $target$ starts with are stored to form the first nodes of the permutation tree. 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 length 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 because it is 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:

$$\operatorname*{arg\,max}_{s\in S} f(s) := \left\{ s \in S : \sum_{i=1}^{|s|} \sqrt[s_i]{\frac{s_i}{t} \div |s|} \right\} \tag{3.1}$$

Where $S$ is a set of segmentation tuples $s$, $|s|$ is the length of the tuple (read: number of segments) and $s_i$ indicates the length of the segment at position $i$ in tuple $s$. For every segment (vertex) in a segmentation tuple (branch) the segment's length is divided by the token length $t$ to get the coverage the morpheme provides towards the target string. The number of segments $|s|$ is a divisor meant to cap the number of segments in a segmentation. It prevents choosing a segmentation overflowing with too many short segments. Short segments are convenient for completing a segmentation, but will increase the chance of slicing a token where it linguistically lacks sense to do so. Lastly the ${s_i}^{\text{th}}$ root implements a bias against segments that are too long. While the accuracy decreases for longer words, this method of maximization performs reasonably well in the range of 1–4 syllable tokens, which make up around 87% of the verb vocabulary.

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 POS- members displays the probable shape of segmentation for a token. As mentioned in 3.2.2, 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

### 3.2.2 Model Training

All fine-tuned models use the bert-base-german-cased (bbgc) baseline by Chan, Schweter, and Möller (2019). For reference and readability they are given shortened IDs as follows:

| ID | Full name |
|---|---|
| mwg5 | mlm_wmt_gpc500k |
| mwo5 | mlm_wmt_oscar500k |
| msg5 | mlm_std_gpc500k |
| mso5 | mlm_std_oscar500k |
| bbgc | bert-base-german-cased |

Table 3: ID references assigned to full model names

The full model names in Table 3 captures broad characteristics of the model. For example, $\boxed{\text{mwo5}}$ is the $\boxed{\text{m}}$asked language model using $\boxed{\text{W}}$ordmap tokenization fine-tuned on the $\boxed{\text{O}}$scar dataset of $\boxed{\text{5}}$00k samples, while msg5 is the the masked language model using standard BERT wordpiece tokenization fine-tuned on the GerParCor dataset of 500k samples. All models except for bbgc are trained on the same parameters as seen in table Table 4.

| ID | Corpus | Tokenization | LR | Steps | Batchsize | WU | Base |
|---|---|---|---|---|---|---|---|
| mwg5 | GerParCor | BWP + WM | 0.0003 | 31250 | 16 | 500 | bbgc |
| mwo5 | Oscar | BWP + WM | 0.0003 | 31250 | 16 | 500 | bbgc |
| msg5 | Gerparcor | BWP | 0.0003 | 31250 | 16 | 500 | bbgc |
| mso5 | Oscar | BWP | 0.0003 | 31250 | 16 | 500 | bbgc |
| bbgc | Mixed | BWP | 0.0001 | 810k/30k | 1024 | 10000 | - |

Table 4: List of all used models and their hyperparameters. LR = learning rate, WU = warmup steps.

The first four models use bbgc as baseline, meaning that they inherit its structural properties. DeepsetAI released bbgc training it in two phases (see column Steps in Table 4) with differing sequence lengths 128/512 respectively. The maximum sequence length for each model is 512, the base vocabulary size is 30k. Models mwg5, mwo5, msg5, and mso5 where trained with less steps and thus higher learning rate to converge quicker with the given datasets. The models where implemented in PyTorch (Paszke et al. (2019)) and trained on the GPUs Quadro RTX 8000 (48Gb) and NVIDIA GeForce GTX 1080 Ti (11Gb) provided at Text Technology Lab (TTLab).

#### 3.2.3 Benchmark

The original draft for this thesis featured a benchmark with a translated multiple choice question answering task from the oLMpics benchmark (Talmor et al. 2020). Due to incompatible versioning dependencies this was not feasible and a substitute had to be found. To test the performance of trained models a one-shot sequence classification task is set up instead. The task consists of ~11k samples containing a sentence and a single label. Each sentence is the title of a news article found on the german Wikinews site, belonging to one of the categories listed on their topic section[1]. Only those categories which appeared at least 200 times where selected for this task, leaving a total of 24 labels to choose from. Every model is then trained and valited on a split (train: 70%, test: 23%, validation: 7%) with a training and evaluation batch size of 16, a less sharp learning rate at 0.00002, and three epochs.

[1]https://de.wikinews.org/wiki/Kategorie:Themenportal

# 4 Results

## 4.1 Benchmark

This section is a comparative description of the results from the sequence classification task described in subsection 3.2.3. For every benchmark of three epochs precision, recall and F1 are given. Precision reflects the amount of correct predictions a model has made for that particular class. Recall shows how often the model correctly predicts a class in relation to all positive predictions. F1 is a score compromising precision and recall through the *harmonic mean* of precision and recall, measuring the models accuracy.

There is a common trend in all runs displaying a growth in precision, recall and F1. This means that all models have continously improved predicting the class of a news title. The highest F1 in the last epoch achieved for any of the smaller models (mso5, msg5, mwo5, mwg5) is mwo5 with a score of 0.69194 (Table 5). Meanwhile, the lowest F1 is found in Table 6 at 0.590542 mwg5. A similar arrangement is found when comparing final test scores in the set of smaller models: mwo5 > mso5 > msg5 > mwg5 ($0.442827 > 0.405168 > 0.392490 > 0.389116$).

| **mwo5** | **Epoch 1** | **Epoch 2** | **Epoch 3** | **Test score** |
|---|---|---|---|---|
| Precision | 0.292614 | 0.446338 | 0.71387 | 0.449735 |
| Recall | 0.329531 | 0.552598 | 0.73384 | 0.474525 |
| F1 | 0.242739 | 0.473851 | 0.69194 | 0.442827 |

Table 5: Metrics for masked language model trained on the Oscar dataset with Wordmap infused tokenization. Evaluated on sequence classification task.

Of all small models, mwo5 has the best results for all three metrics precision, recall and F1. It seemingly learns the fastest out of the four, starting with a recall of 0.329521 and ending on 0.73384 on the last epoch, overtaking its standard counterpart mso5 at the second epoch.

Contrary to the pattern of mwo5, the wordmap model trained on GerParCor data predicts less accurately than the standard msg5 model. The inital epoch shows the lowest scores of all models for precision, recall and logically F1. None of the predictions keep up with the accuracy of the other small model at any point in training.

| mwg5 | Epoch 1 | Epoch 2 | Epoch 3 | Test score |
|---|---|---|---|---|
| Precision | 0.237664 | 0.399781 | 0.603534 | 0.441891 |
| Recall | 0.244613 | 0.463878 | 0.637516 | 0.440304 |
| F1 | 0.163024 | 0.389905 | 0.590542 | 0.389116 |

Table 6: Metrics for masked language model trained on the GerParCor dataset with Wordmap infused tokenization. Evaluated on sequence classification task.

| mso5 | Epoch 1 | Epoch 2 | Epoch 3 | Test scores |
|---|---|---|---|---|
| Precision | 0.269615 | 0.422096 | 0.596987 | 0.395879 |
| Recall | 0.351077 | 0.501901 | 0.657795 | 0.446388 |
| F1 | 0.266260 | 0.412598 | 0.604824 | 0.405168 |

Table 7: Metrics for masked language model trained on the GerParCor dataset with bbgc tokenization. Evaluated on sequence classification task.

With an F1 of 0.604824 at the final epoch mso5 is the lowest scoring standard model, despite starting with similar scores in epoch 1 (Table 7 vs. Table 8). The standard F1 test scores are close to each other, deviating only by one second decimal point indicating comparable benchmark performance.

| msg5 | Epoch 1 | Epoch 2 | Epoch 3 | Test scores |
|---|---|---|---|---|
| Precision | 0.297466 | 0.517110 | 0.656808 | 0.439873 |
| Recall | 0.359949 | 0.544994 | 0.676806 | 0.439924 |
| F1 | 0.267111 | 0.480420 | 0.626593 | 0.392490 |

Table 8: Metrics for masked language model trained on the GerParCor dataset with bbgc tokenization. Evaluated on sequence classification task.

All small model scores are relatively close to each other, but ultimately outperformed by the baseline model.

## 4.2 Tokenization

Show specific examples of tokenization and analyze them qualitatively (maybe quantitatively)

wordmap: range around the median but

| bbgc | Epoch 1 | Epoch 2 | Epoch 3 | Test scores |
|---|---|---|---|---|
| Precision | 0.646150 | 0.768675 | 0.860180 | 0.622436 |
| Recall | 0.709759 | 0.804816 | 0.883397 | 0.637262 |
| F1 | 0.660588 | 0.778371 | 0.868166 | 0.624789 |

Table 9: Metrics for masked language model baseline bert-base-german-cased[1]. Evaluated on sequence classification task.

| Summary | bbgc | std+oscar | std+gpc | wmt+oscar | wmt+gpc |
|---|---|---|---|---|---|
| Precision | **0.622436** | 0.395879 | 0.439873 | 0.441891 | 0.449735 |
| Recall | **0.637262** | 0.446388 | 0.439924 | 0.440304 | 0.474525 |
| F1 | **0.624789** | 0.405168 | 0.392490 | 0.389116 | 0.442827 |

Table 10: Test score summary for all evaluated models.

# 5 Discussion

# 6 Conclusion

# 7 Testchapter

“This is a quote by textquote” (DeepL 2021) “This is a quote by enquote”

# 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.

Aepli, Noëmi and Rico Sennrich (2021). *Improving Zero-shot Cross-lingual Transfer between Closely Related Languages by injecting Character-level Noise.* DOI: 10.48550/ARXIV.2109.06772. URL: https://arxiv.org/abs/2109.06772.

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.

Bentz, Christian, Tatyana Ruzsics, Alexander Koplenig, and Tanja Samardžić (Dec. 2016). "A Comparison Between Morphological Complexity Measures: Typological Data vs. Language Corpora". In: *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC).* Osaka, Japan: The COLING 2016 Organizing Committee, pp. 142–153. URL: https://aclanthology.org/W16-4117.

Bouckaert, Remco et al. (2012). "Mapping the Origins and Expansion of the Indo-European Language Family". In: *Science* 337.6097, pp. 957–960. DOI: 10.1126/science.1219669. eprint: https://www.science.org/doi/pdf/10.1126/science.1219669. URL: https://www.science.org/doi/abs/10.1126/science.1219669.

Brennan, J.R. (2022). *Language and the Brain: A Slim Guide to Neurolinguistics.* Oxford linguistics. Oxford University Press. ISBN: 9780198814757. URL: https://books.google.de/books?id=KLe5zgEACAAJ.

Chan, Branden, Stefan Schweter, and Timo Möller (June 2019). *GermanBERT.* URL: https://huggingface.co/bert-base-german-cased.

Colman, Andrew M. (2009). *morpheme.* DOI: 10.1093/acref/9780199534067.013.5219. URL: https://www.oxfordreference.com/view/10.1093/acref/9780199534067.001.0001/acref-9780199534067-e-5219.

Çöltekin, Çağrı and Taraka Rama (2022). "What do complexity measures measure? Correlating and validating corpus-based measures of morphological complexity". In: *Linguistics Vanguard*. DOI: doi:10.1515/lingvan-2021-0007. URL: https://doi.org/10.1515/lingvan-2021-0007.

Comrie, Bernard (1989). "Morphological Typology". In: *Language universals and linguistic typology*. 2nd ed. University of Chicago Press, pp. 42–56.

Creutz, Mathias and Krista Lagus (2002). "Unsupervised Discovery of Morphemes". In: *CoRR* cs.CL/0205057. URL: https://arxiv.org/abs/cs/0205057.

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

Delogu, Francesca, Harm Brouwer, and Matthew W. Crocker (2019). "Event-related potentials index lexical retrieval (N400) and integration (P600) during language comprehension". In: *Brain and Cognition* 135, pp. 1–3. ISSN: 0278-2626. DOI: https://doi.org/10.1016/j.bandc.2019.05.007. URL: https://www.sciencedirect.com/science/article/pii/S0278262618304299.

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.

Ehret, Katharina, Alice Blumenthal-Dramé, Christian Bentz, and Aleksandrs Berdicevskis (2021). "Meaning and Measures: Interpreting and Evaluating Complexity Metrics". In: *Frontiers in Communication* 6. ISSN: 2297-900X. DOI: 10.3389/fcomm.2021.640510. URL: https://www.frontiersin.org/articles/10.3389/fcomm.2021.640510.

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.

Hofmann, Valentin, Hinrich Schuetze, and Janet Pierrehumbert (May 2022). "An Embarrassingly Simple Method to Mitigate Undesirable Properties of Pretrained Language Model Tokenizers". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Dublin, Ireland: Association for Computational Linguistics, pp. 385–393. DOI: 10.18653/v1/2022.acl-short.43. URL: https://aclanthology.org/2022.acl-short.43.

Iacobini, Claudio and Francesca Masini (Sept. 2005). "Verb-particle Constructions and Prefixed Verbs in Italian: Typology,Diachrony and Semantics". In: *Proceedings of the Fifth Mediterranean Morphology Meeting (MMM5)*. Università degli Studi di Bologna. URL: https:

//www.academia.edu/1183339/Verb_particle_constructions_and_prefixed_verbs_in_Italian_typology_diachrony_and_semantics.

Jianlong Zhou, Fang Chen (2018). *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent*. 1st ed. 2018. Human–Computer Interaction Series. Springer. ISBN: 3319904027; 9783319904023. URL: libgen.li/file.php?md5=1c5a5ce9773ae7e1c0c6fa04da9e0c03.

Kimppa, Lilli et al. (2019). "Acquisition of L2 morphology by adult language learners". In: *Cortex* 116. Structure in words: the present and future of morphological processing in a multidisciplinary perspective, pp. 74–90. ISSN: 0010-9452. DOI: https://doi.org/10.1016/j.cortex.2019.01.012. URL: https://www.sciencedirect.com/science/article/pii/S0010945219300310.

Kiss, Katalin É. and Veronika Hegedus, eds. (2021). *Postpositions and Postpositional Phrases*. Amsterdam: Amsterdam University Press, p. 262. ISBN: 9789048544608. DOI: doi:10.1515/9789048544608. URL: https://doi.org/10.1515/9789048544608.

Kühl, Niklas, Marc Goutier, Lucas Baier, Clemens Wolff, and Dominik Martin (2020). "Human vs. supervised machine learning: Who learns patterns faster?" In: *CoRR* abs/2012.03661. arXiv: 2012.03661. URL: https://arxiv.org/abs/2012.03661.

Lehman, Christian W. (Aug. 2022). *Indogermanisch*. URL: https://www.christianlehmann.eu/ling/sprachen/indogermania/RomGesch/idg.php.

Nikanne, Urpo (Dec. 2017). *Finite sentences in Finnish: Word order, morphology, and information structure*. DOI: 10.5281/zenodo.1117710. URL: https://doi.org/10.5281/zenodo.1117710.

Ortiz Su'arez, Pedro Javier, Laurent Romary, and Benoit Sagot (June 2020). "A Monolingual Approach to Contextualized Word Embeddings for Mid-Resource Languages". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 1703–1714. URL: https://www.aclweb.org/anthology/2020.acl-main.156.

Ortiz Su'arez, Pedro Javier, Benoit Sagot, and Laurent Romary (2019). "Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures". en. In: ed. by Piotr Bański et al. Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019. Mannheim: Leibniz-Institut f"ur Deutsche Sprache, pp. 9–16. DOI: 10.14618/ids-pub-9021. URL: http://nbn-resolving.de/urn:nbn:de:bsz:mh39-90215.

OSCAR (Jan. 2022). *The OSCAR project (Open Super-large Crawled Aggregated coRpus)*. URL: https://oscar-project.org/.

Paszke, Adam et al. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. DOI: 10.48550/ARXIV.1912.01703. URL: https://arxiv.org/abs/1912.01703.

Peters, Ben and Andre F. T. Martins (July 2022). "Beyond Characters: Subword-level Morpheme Segmentation". In: *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Seattle, Washington: Association for Computational Linguistics, pp. 131–138. DOI: 10.18653/v1/2022.sigmorphon-1.14. URL: https://aclanthology.org/2022.sigmorphon-1.14.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). "Improving language understanding by generative pre-training". In: URL: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

Rust, Phillip, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych (2020). *How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models*. DOI: 10.48550/ARXIV.2012.15613. URL: https://arxiv.org/abs/2012.15613.

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.

Schwartz, Lane et al. (2020). *Neural Polysynthetic Language Modelling*. DOI: 10.48550/ARXIV.2005.05477. URL: https://arxiv.org/abs/2005.05477.

Sennrich, Rico, Barry Haddow, and Alexandra Birch (Aug. 2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://aclanthology.org/P16-1162.

Talmor, Alon, Yanai Elazar, Yoav Goldberg, and Jonathan Berant (Dec. 2020). "oLMpics-On What Language Model Pre-training Captures". In: *Transactions of the Association for Computational Linguistics* 8, pp. 743–758. ISSN: 2307-387X. DOI: 10.1162/tacl_a_00342. eprint: https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00342/1923716/tacl_a_00342.pdf. URL: https://doi.org/10.1162/tacl%5C_a%5C_00342.

Toraman, Cagri, Eyup Halit Yilmaz, Furkan Şahinuç, and Oguzhan Ozcelik (2022). *Impact of Tokenization on Language Models: An Analysis for Turkish*. DOI: 10.48550/ARXIV.2204.08832. URL: https://arxiv.org/abs/2204.08832.

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

Wartena, Christian (2019). “A Probabilistic Morphology Model for German Lemmatization”. en. In: Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019), pp. 40–49. DOI: 10.25968/opus-1527. URL: http://nbn-resolving.de/urn:nbn:de:bsz:960-opus4-15271.

Wiktionary (Sept. 2022). *Verzeichnis: Deutsch.* URL: https://de.wiktionary.org/wiki/Verzeichnis:Deutsch.

Wu, Yonghui et al. (2016). *Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.* DOI: 10.48550/ARXIV.1609.08144. URL: https://arxiv.org/abs/1609.08144.