

# Tik-to-Tok: Translating Language Models One Token at a Time: An Embedding Initialization Strategy for Efficient Language Adaptation

Anonymous ACL submission

## Abstract

Training monolingual language models for low and mid-resource languages is made challenging by limited and often inadequate pretraining data. In this study, we propose a novel model conversion strategy to address this issue, adapting high-resources monolingual language models to a new target language. By generalizing over a word translation dictionary encompassing both the source and target languages, we map tokens from the target tokenizer to semantically similar tokens from the source language tokenizer. This one-to-many token mapping improves tremendously the initialization of the embedding table for the target language. We conduct experiments to convert high-resource models to mid- and low-resource languages, namely Dutch and Frisian. These converted models achieve a new state-of-the-art performance on these languages across all sorts of downstream tasks. By reducing significantly the amount of data and time required for training state-of-the-art models, our novel model conversion strategy has the potential to benefit many languages worldwide.

## 1 Introduction

Large pre-trained language models have emerged as a standard approach in NLP (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020). Unfortunately, high-quality monolingual models exist only for a handful of languages. Worse, they are rarely kept up-to-date because of the cost of retraining.

Multilingual models, often touted as a solution, have their own challenges: interference between languages, poor tokenization, large model size, and more (Wang et al., 2020; Pires et al., 2019). Moreover, translating inputs into English and using a monolingual English model for inference can sometimes surpass the performance of a fine-tuned multilingual model (Artetxe et al., 2023).

A more promising strategy seems to be model conversion. In this approach, the original tokenizer of an existing monolingual model is discarded in favor of an entirely new vocabulary, adapted to the target language; tokens shared between the two tokenizers keep their existing embedding, while newly-introduced tokens are randomly initialized (Artetxe et al., 2020a; de Vries et al., 2021; Garcia et al., 2021; Gogoulou et al., 2022a).

While tokenizer upgrades for language models are nothing new, existing approaches only consider tokens as black boxes, ignoring the character strings they represent, as well as their semantics. They are thus ill-equipped to deal with languages which form multiword compounds, such as Dutch or German. In these languages, compound words can be formed by agglutinating several existing words together (e.g. *corporate credit* translates to the compound *bedrijfskrediet* in Dutch).

Compounds complicate the use of the popular merge-based tokenizers, as letters at subword boundaries often get merged to the incorrect segment, resulting in several partial tokens with extra or missing letters, all representing the same concept but in different compounds (see Table 1). This reduces the amount of training data which each of these tokens receives during training, lowering the quality of their representation, thereby occupying precious space in the embedding table for low-quality tokens. This also makes them difficult to align with existing source language tokens using the trivial mapping strategies described thus-far.

In this work, we address this issue by initializing the embedding of each token of a new target vocabulary using a weighted combination of embeddings of similar tokens from the source model, the discovery of which consists of a novel “token translation” task which relies on the character composition and semantics of these tokens, which we approximate using the character n-grams these tokens contain.

| TOKENS WITH EXTRA LETTERS  |   |
|--|---|
| $E_t[\dots\text{ingsbedrijf}] =$<br>+ 0.5 * $E_s[\dots\text{company}]$<br>+ 0.3 * $E_s[\dots\text{Company}]$<br>+ 0.2 * $E_s[\text{Company}]$        | $E_t[\text{universiteits}\dots] =$<br>+ 0.5 * $E_s[\text{university}]$<br>+ 0.3 * $E_s[\text{University}]$<br>+ 0.2 * $E_s[\dots\text{University}]$ |
| TOKENS WITH MISSING LETTERS  |   |
| $E_t[\dots\text{oeding}] =$<br>+ 0.5 * $E_s[\dots\text{feeding}]$<br>+ 0.3 * $E_s[\dots\text{eding}]$<br>+ 0.2 * $E_s[\dots\text{breeding}]$         | $E_t[\text{inschrijf}\dots] =$<br>+ 0.5 * $E_s[\text{inscribed}]$<br>+ 0.3 * $E_s[\text{inserting}]$<br>+ 0.2 * $E_s[\text{inline}]$                |
| TOKENS WITH NOVEL SPLITS   |   |
| $E_t[\text{Administr}\dots] =$<br>+ 0.5 * $E_s[\text{Administ}\dots]$<br>+ 0.3 * $E_s[\dots\text{Administ}]$<br>+ 0.2 * $E_s[\text{administr}\dots]$ | $E_t[\text{Afrik}\dots] =$<br>+ 0.5 * $E_s[\text{Afric}\dots]$<br>+ 0.3 * $E_s[\text{Africa}]$<br>+ 0.2 * $E_s[\text{African}]$                     |
| TOKENS FOR WORD ENDINGS  |   |
| $E_t[\dots\text{ventie}] =$<br>+ 0.5 * $E_s[\dots\text{vention}]$<br>+ 0.3 * $E_s[\dots\text{inence}]$<br>+ 0.2 * $E_s[\text{invention}]$            | $E_t[\dots\text{geerd}] =$<br>+ 0.5 * $E_s[\dots\text{inated}]$<br>+ 0.3 * $E_s[\dots\text{urized}]$<br>+ 0.2 * $E_s[\dots\text{itized}]$           |

Table 1: Categories of Dutch tokens not covered by a translation dictionary, and their corresponding English tokens based on the Tik-to-Tok strategy.

We hypothesize and show that this strategy yields considerable advantages over training a language model from scratch, with regards to both training efficiency and downstream performance. In the next chapters, we develop the following two key contributions:

- We propose a novel cross-lingual model conversion strategy for low-resource languages that does not require further pre-training on a downstream corpus, by leveraging a translation dictionary instead of a corpus (§ 4.1).
- We show that, even for mid-resource languages for which enough data exists for training dedicated language models, applying our strategy and finetuning on the available corpus performs better than training such a language model from scratch (§ 4.2).

In the next section, we introduce the state of the art strategies for cross-lingual transfer learning and tokenizer upgrades. We then follow up with a more comprehensive description of our methodology in section 3, and showcase our results on Frisian and Dutch model conversion in section 4.

## 2 Background and Related Work

Adapting existing language models to new languages –also called model conversion– would be highly desirable to reduce the cost of language models. Indeed, while large language models perform exceptionally well on many downstream tasks, this performance is dependent on the high computational cost and data requirements of model pre-training, which not every language can afford.

Unfortunately, the usage of different vocabularies and tokens prevent the direct transfer of embedding weights during model conversion, requiring a random reinitialization of all or most embeddings, resulting in models of lower quality than those trained from scratch on those languages, or benefiting from multilingual transfer learning.

Additionally, encoding input sentences using a subword tokenizer has become the dominant paradigm embraced by the state-of-the-art language models adapted from the Transformer architecture (Vaswani et al., 2017). These tokenizers rely on a greedy approach to assign tokens to the most frequently occurring subwords by iteratively merging frequent substring pairs to form new tokens (Sennrich et al., 2016; Wu et al., 2016).

Encoding words as substring tokens is undoubtedly a trade-off, whose challenges are by now well-known (Provilkov et al., 2020; Rogers et al., 2020). One such challenge pertains to the non-morphological segmentation of compound words, as highlighted by Vilar and Federico (2021).

Before explaining how we address these shortcomings, let us consider the existing strategies for low-resources languages as well as relevant work pertaining to tokenizer and model conversion.

As previously mentioned, the usage of multilingual language models remains the most common strategy used to deal with low-resource languages. Low-resources languages in multilingual models benefit from both the joint training with languages with higher-quality resources, and from a knowledge transfer obtained through to the use of a shared vocabulary (Pires et al., 2019).

One drawback of this approach is the large vocabulary it requires to properly encode all these languages, leading to an embedding table which is significantly larger than for monolingual models.

Additionally, due to language interference, monolingual performance of mid-resourced languages only increases up to a point as more languages are added to the training, after which it

starts to degrade (Conneau et al., 2020). The usage of multilingual language models remains therefore a trade-off for many languages worldwide, which could theoretically benefit from smaller and more powerful models, but might not do in practice due to lack of training strategies that are data-efficient enough to warrant the cost.

The perceived significance of this phenomenon seems to be increasing, as Artetxe et al. (2023) have recently demonstrated that meticulously fine-tuned multilingual model could not attain the same performance as similarly-sized monolingual English models used on translated inputs. A possible hypothesis is that it might not be possible to eliminate entirely the language interference in multilingual models, even after a monolingual finetuning.

Instead of translating inputs into English, a more sensible approach would be to convert English models into models for the target language. The first forays into the realms of model conversion originated from Artetxe et al. (2020b), who illustrated

that performing cross-lingual transfers of monolingual models was possible, although at a significant performance cost. Concretely, after training a new tokenizer for a target language, the authors trained a new embedding table from scratch for the new target tokens, after freezing the attention layers.

A similar approach was successfully applied on Dutch to adapt BERT (de Vries et al., 2021) and GPT-2 (de Vries and Nissim, 2021), with varying degrees of success. Their and other works demonstrated that reusing the embeddings of shared tokens could already reduce the performance gap between converted and from-scratch models.

Further investigations by Gogoulou et al. (2022b) demonstrated that cross-lingual model conversion can, in some cases, be beneficial to model performance even in the original language, after back-conversion of the target model. This seemed to indicate it should in theory be possible to produce state-of-the-art models through conversion, if a suitable model conversion strategy was devised.

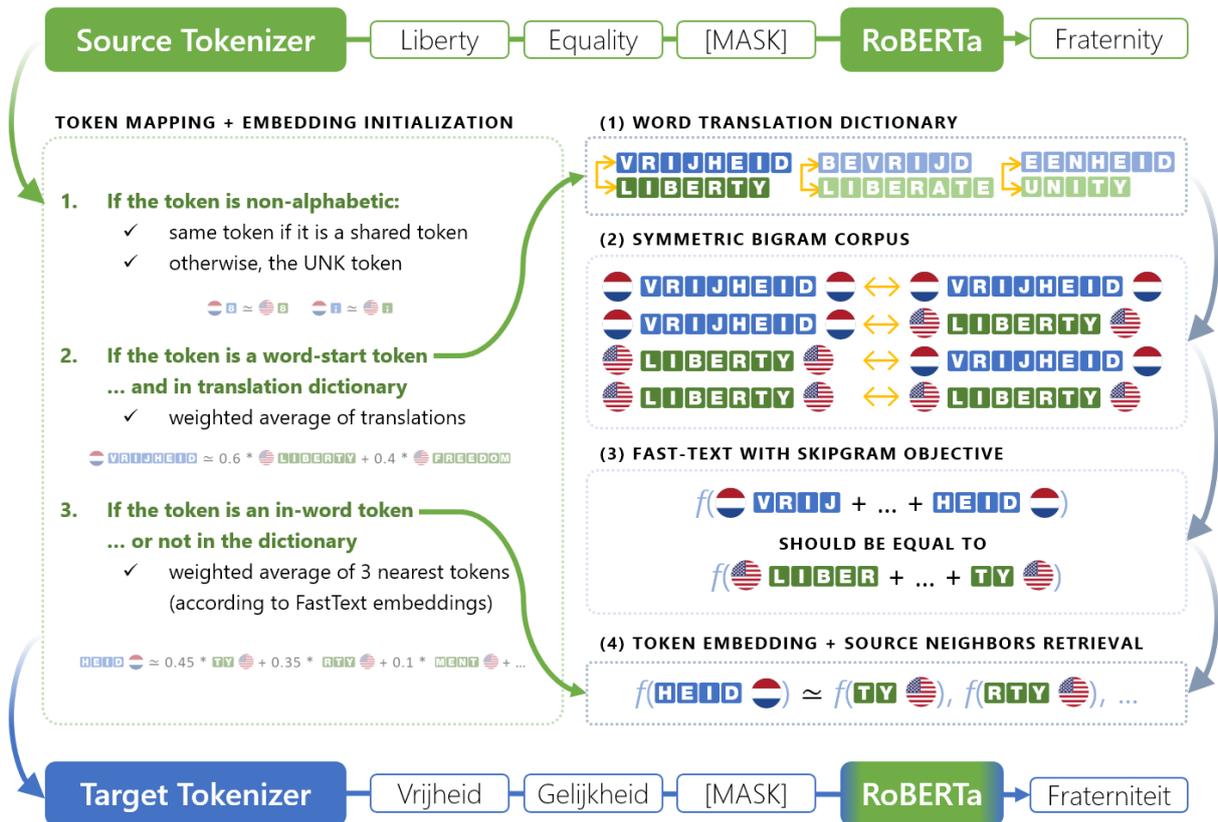


Figure 1: Illustration of the model conversion process. While non-alphabetical tokens usually have an exact match in the source tokenizer, we map the other tokens either using the word translation dictionary (1), if possible, or using a character n-gram embedding model that is able to generalize to partial tokens (4). To ensure a proper learning of the character n-gram embedding model, we generate then train on a symmetric bigram corpus (2), where every word is equally likely to be paired with itself or its translation, ensure that the skipgram distribution of a word and its translation are identical, imposing a strong equality constraint on their summed n-gram embeddings (3).

### 3 Methodology

In this work, we introduce a new state-of-the-art strategy (Tik-To-Tok) to convert an existing monolingual language model from the language it was originally pretrained on (called the source language) to a new language of interest (called target). We achieve this by replacing its current tokenizer and token embedding table by new ones, which are better suited for the target language or domain.

In line with previous works, we achieve this by reusing entries of the old embedding table to initialize the new embedding table. However, unlike previous works, we rely on a word translation dictionary spanning the source and target languages to compute a token mapping between the source and target embedding tables.

As was noted in the previous section, a better initialization of the new embedding table helps to better preserve the downstream performance of the model post conversion. However, not every shared token makes sense to reuse across languages. While some words, such as “*computer*” share the same meaning in Dutch and English, some other shared words such as “*of*” have different meanings in the two languages.

Using a word translation dictionary to perform the initialization is therefore expected to bring benefits, first by enabling many more tokens to be initialized (since being spelled identically is no longer a requirement for a better initialization), but also by reducing the frequency of inaccurate initialization where words are incorrectly initialized with the embedding of false friends.

Additionally, to address the issue of subword tokens (which are not contained as-is in a word-based translation dictionary), we devise a fallback strategy relying on an estimation of the semantic similarity of tokens. We hypothesize that it is possible to approximately translate even partial subword tokens, using an estimation of their semantic meaning based on the character n-grams they contain.

For this, we rely on fastText (Mikolov et al., 2018), a word embedding strategy capable of producing embeddings for out-of-vocabulary subwords by summing the embeddings of the character n-gram they contain. The main advantage of fastText is its robustness against typos, according to the authors. Using fastText, we aim to provide an approximate token mapping (e.g. from *Tik* to *Tok*) based on the insights of the translation dictionary, even for out-of-vocabulary or for subword tokens.

To this end, we develop a bilingual fastText embedding model that generalizes over the word translation dictionary, in order to provide a mapping for all tokens not covered by the dictionary itself. We do this by embedding all source and target tokens in a shared fastText space and, for each token of the target language, by retrieving the tokens of the source which are its closest neighbors.

Irrespective of the chosen strategy, we then initialize the new embedding table for each token as a weighted average of its “translations”. We put additional emphasis on the first and second best matches, which we found to be of higher quality. To do so, we assign 30% of the weight to the best match, 10% to the second best match, and divide the remaining 60% equally among all candidates.

We aim to show that this novel token translation strategy extends by a lot the potential of the translation dictionary, since many tokens (especially in Germanic languages) consist of only a part of a word or compound. We do this by comparing our strategy with state-of-the-art baselines.

#### 3.1 Symmetrization of the dictionary

To compute fastText embeddings with the desired properties, we leverage a well-known property of skip-gram models: words sharing similar neighbor distributions tend to exhibit analogous representations. Using this insight, we transform the word translation dictionary into a bigram corpus featuring a symmetric distribution (i.e., where each word is paired equally often to itself and to its translation in the other language; see Figure 1.2).

That way, a word and its translation feature exactly the same distribution of neighbors (50% themselves, and 50% their translation; at least for words that are perfect translations of each other) and the sum of embeddings of their character-n-grams are encouraged to match as a result (see Figure 1.3).

To differentiate words from the source and target languages, a language-specific tag is prepended and appended to every word in the dictionary (we represent these with flags in Figure 1, but use simpler unicode characters in our implementation).

For languages which form multiword compounds, like Frisian and Dutch, we perform an additional data augmentation by adding copies of each word where either the start or end tags are omitted, to simulate partial compounds. We rely on fastText to deal with extra or missing letters.

### 3.2 FastText training

Once the symmetric bigram corpus described above is generated, we train a fastText character n-gram model on the word pairs it contains. This fastText model will be used to compute approximate semantic representations of tokens of the target language which are not present in the dictionary itself, and identify semantically similar tokens in the source language. In this section, we describe the hyperparameters used for training of the fastText model.

The default fastText training makes use of the skip-gram training objective, and we keep this default configuration in our approach. However, given that all training examples consist of pairs of words only, the skip-gram objective becomes equivalent to a neighbor prediction objective. As a result, the window size parameter no longer plays a role, and does not have to be optimized.

To construct the embeddings of the words in the translation dictionary, fastText enumerates all the character n-grams they contain whose length is contained between `MinN` and `MaxN` characters. Each of these n-grams is then assigned an embedding, and the sum of these embeddings is used to represent the word. The default `MinN` and `MaxN` values of fastText embed all n-grams that are 3-6 characters long, but a shifted range of 4-7 characters was used in our experiments, based on the findings that 3-grams are counter-productive both for the Dutch and English languages, as was also evidenced by Novotný et al. (2021).

In our experiments, we train 64-dimensional fastText embeddings on the synthetic corpus, for 5 epochs. We do not find these specific parameters to have a large impact on the resulting mapping, although training for 10 epochs or more seemed to result in an increased overfitting.

All words from both languages are included in the fastText output space, as required by the neighbor prediction objective. All the remaining hyperparameters are set to their default value.

### 3.3 Token matching

Once the fastText model is trained, we have all the tools necessary to perform our token mapping.

**For non-alphabetic target tokens:** we reuse the embeddings of tokens from the source language for tokens that are shared between the two languages, without change; this mainly concerns punctuation and numeric tokens. For tokens that are not shared, the special UNK token is used as a fallback.

**For tokens present in the dictionary:** we rely on the set of translations found in the dictionary. After sorting the translations of the target word by frequency, we return the weighted average of the embeddings of all words from the list. For words in that list that do not have a corresponding token in the source tokenizer, we fallback to first token of their tokenization by the source tokenizer).

**For the remaining tokens:** we calculate their fastText embedding (after eventually prepending and/or appending the language tags, as appropriate for that token) and retrieve the three best matching source tokens using cosine similarity as a metric, and average the embedding of the three tokens.

The strategy described above will frequently map target tokens to three or more source tokens. To account for the higher quality of top-ranking matches, we pre-allocate 30% of the weight to the highest-ranking match and 10% to the second highest-ranking match, with the residual 60% being evenly divided among all candidates (including the first two). Alternatives for the weighting scheme can be investigated in future works.

Because Tik-to-Tok initializes all embeddings using the source model, our initialization strategy does not call for a random initialization unlike the previous state-of-the-art strategies.

### 3.4 Embedding finetuning

While the extensive work above yields an excellent initialization for the new embedding table of the transformer, difference in linguistic patterns between the source and target language result in a higher masked language modeling (MLM) loss in the target language after reinitialization.

Finetuning the newly-initialized embeddings on a corpus in the target language can be used to reduce the loss again, by learning new patterns. To prevent catastrophic forgetting during that phase of the training, all the other parameters of the Transformer model are kept frozen.

### 3.5 Model finetuning

Once the the embeddings have converged and the MLM loss stabilizes, the base model can optionally be unfreezed in order to continue the training with the remaining data. During this second phase of the training, a lower learning rate is recommended, and a longer warmup. We provide hyperparameters for the finetuning stages in Appendix A. The benefits brought by further finetuning of the whole model are analyzed in more details in § 4.2.

## 4 Experimental Evaluation

To evaluate our model conversion strategy, we conduct experiments in the low-resource and mid-resource realms of language model conversion. The goal of these experiments is to establish the supremacy of our proposed Tik-to-Tok initialization strategy, and benchmark its post-finetuning performance against state-of-the-art monolingual as well as state-of-the-art multilingual models.

### 4.1 Low-resource Languages

To compare our embedding table initialization strategy with state-of-the-art approaches, we first focus on experiments on a low-resource language: Frisian, a West Germanic language spoken by about 400,000 native speakers, mostly located in the province of Friesland (Fryslân) in the north of the Netherlands. The grammar of the Frisian language is similar to other West Germanic languages and most notably the Dutch language.

The Oscar 2019 corpus contains only 35 megabytes of data for Frisian, making it impossible to training large language models for Frisian. However, its proximity with Dutch enabled [de Vries et al. \(2021\)](#) to convert an existing Dutch model into a mostly functional Frisian model.

In this work, we take advantage of the very small size of the Frisian corpus used in that experiment to quantify the impact of different embedding initialization strategies on the conversion outcome.

In particular, we investigate four initialization strategies: **[1]** a complete re-initialization of the embedding table (baseline), **[2]** a re-utilization of the source embeddings of all tokens which are shared between the Dutch and Frisian tokenizers (previous state of the art), **[3]** a simplified dictionary mapping strategy where only non-alphabetic tokens and full-word tokens present in the word translation dictionary are initialized based on the source embeddings, and finally **[4]** our complete Tik-To-Tok mapping strategy where gaps in the dictionary

| Embedding Initialization Strategy             | 0ep.        | 1ep. | 2ep.        |
|---|-------------|------|-------------|
| <b>[1]</b> Random initialization (Baseline)   | 9.11        | 7.11 | 7.06        |
| <b>[2]</b> Mapping shared tokens (2023 SotA)  | 7.34        | 5.22 | <u>5.02</u> |
| <b>[3]</b> Mapping with dictionary (Ablation) | 6.56        | 4.45 | 4.23        |
| <b>[4]</b> Mapping with dictionary+fastText   | <u>5.50</u> | 4.00 | <b>3.79</b> |

Table 2: MLM loss of our Dutch model after conversion into a Frisian model in function of the chosen token embedding initialization strategy (after 0, 1, and 2 epochs).

mapping are filled using the fastText embeddings computed for source and target tokens.

In this experiment, we convert our *roberta-large-nl-oscar19* model (described in the next section) into a Frisian model using these various strategies. We therefore apply our mappings between the Dutch tokenizer of RobBERT ([Delobelle et al., 2020](#)), and the Frisian tokenizer devised by [de Vries et al. \(2021\)](#).

The model parameters are kept frozen throughout our experimentation, to the exception of the language modeling head and the embedding table. The limited scope of the parameter training is justified by the data scarcity, the high grammatical proximity between the Frisian and Dutch languages, and the strong cultural and geographical ties between the two communities.

To create a word translation dictionary suitable for the token mapping strategies that require it, and given that the Frisian language is not supported by the OpenSubtitles2018 dataset ([Lison et al., 2018](#)), we combine two manually curated word translation dictionaries between Frisian and Dutch, from respectively [Duijff et al. \(2008\)](#) and [Zanema \(1984\)](#).

We evaluate the models based on the MLM loss after 0, 1 and 2 training epochs on the oscar corpus. This evaluation strategy was chosen to show both differences in post-initialization quality (0 epoch) and the extent to which this difference can be recovered through finetuning using the available Frisian corpus. The loss after 2 training epochs is reported to demonstrate that a plateau is already reached after 1 training epoch, and that little further performance gain is to be expected beyond that point (without providing significantly more training data). Our results are reported in Table 2.

From these results, we draw the following conclusions: **[a]** Irrespective of the conversion strategy, finetuning on the native corpus showed limited ( $\sim 2p$ ) capacity for recovering from the impairment caused by the conversion, highlighting the need of a good initialization. **[b]** Reusing shared tokens is a massive improvement over the random re-initialization of the embedding table, but the final performance remains subpar. **[c]** Reusing embeddings through a dictionary mapping, on the other hand, results in a much better model at initialization; the MLM loss of our Tik-to-Tok model is already comparable with the loss of a finetuned model converted using the SOTA approach (with MLM finetuning bringing additional improvements on top of that).

Table 3: Results for the benchmark by [Delobelle et al. \(2020\)](#) on Natural Language Inference, Sentiment Analysis, Named Entity Recognition, and Part-of-Speech tagging, as well as the pseudo-perplexity (PPL, [Salazar et al., 2019](#)) on the new OSCAR-2023-01 corpus. Converted models use the same tokenizer and corpus as RobBERT.

| CONFIGURATION   |  |        | BENCHMARK SCORES |             |             |             |            |
|---|--|--------|------------------|-------------|-------------|-------------|------------|
| Lang.   | Model  | Params | NLI              | SA          | NER         | POS         | PPL        |
|  | BERTje ( <a href="#">de Vries et al., 2019</a> )   | 109 M  | 83.9             | 93.0        | 88.3        | 96.3        | 33.8       |
|  | RobBERT ( <a href="#">Delobelle et al., 2020</a> ) | 116 M  | 84.2             | 94.4        | <u>89.1</u> | <u>96.4</u> | 13.1       |
|  | <b>Converted camembert-base</b>                    |        |                  |             |             |             |            |
|   | Tik-to-Tok + full finetuning                       | 116 M  | 85.3             | <u>95.8</u> | 84.9        | 94.4        | 12.4       |
|  | <b>Converted gbert-base</b>                        |        |                  |             |             |             |            |
|   | Tik-to-Tok + full finetuning                       | 116 M  | 85.5             | 95.0        | 86.3        | 95.3        | 10.2       |
|  | <b>Converted olm-base</b>                          |        |                  |             |             |             |            |
|   | Tik-to-Tok only (no LM head)                       | 116 M  | 85.0             | 95.5        | 78.6        | 93.8        | $\infty$   |
|   | Tik-to-Tok + embeddings ft.                        | 116 M  | 85.4             | 95.6        | 86.0        | 95.1        | 9.9        |
|   | Tik-to-Tok + full finetuning                       | 116 M  | <u>86.6</u>      | 95.4        | 87.6        | 95.8        | <u>5.9</u> |
|  | <b>Converted roberta-large</b>                     |        |                  |             |             |             |            |
|   | Tik-to-Tok + full finetuning                       | 345 M  | <b>89.2</b>      | <b>97.0</b> | <b>89.5</b> | 96.0        | <b>4.9</b> |
|  | XLM-RoBERTa large (XLM-R)                          | 560 M  | 87.9             | 96.5        | <b>89.5</b> | <b>96.9</b> | 5.5        |

## 4.2 Mid-resource Languages

Even whenever enough data is available to pre-train a language model, our token translation approach can significantly reduce the time and cost required for pre-training such a language model. We demonstrate this by applying our Tik-to-Tok model conversion to Dutch, a mid-resource language.

We train a series of Dutch models, on the same corpus (Oscar19 NL) and using the same tokenizer as RobBERT ([Delobelle et al., 2020](#)), then evaluate them following the same evaluation as this existing Dutch model. We also compare them with BERTje ([de Vries et al., 2019](#)) and the (larger) multilingual XLM-R model ([Conneau et al., 2020](#)).

We evaluate a set of high-resource languages (French, German, and English) to initialize our Dutch models. More specifically we evaluate converted versions of the French CamemBERT-base ([Martin et al., 2020](#)), the German GBERT-base ([Chan et al., 2020](#)), and the English olm-base ([Thrush and Oblokulov, 2022](#)) and RoBERTa-large ([Liu et al., 2019](#)) models. Three different tokenizers (BertTokenizer, RobertaTokenizer, CamembertTokenizer) are covered by these four models, thereby proving that our approach can be used across a wide range of tokenizer implementations.

The models are evaluated on 5 tasks: Sentiment Analysis (SA), Named Entity Recognition (NER), Part-of-Speech tagging (POS), Natural Language Inference (NLI), as well as their Pseudo-Perplexity (PPL) on the recently-crawled Oscar2023 corpus.

To disentangle the effects of the embedding table initialization from the subsequent MLM finetuning, we perform an ablation study where we test three variations of the training setup. Due to the limited impact, this seemed to have on the downstream tasks, we only performed this analysis for our best 116M-parameters model, initialized from olm-base. Results for the other models are presented only after finetuning the weights of the entire model.

In all cases, we train the models following the same procedure: Firstly, we reinitialize the embeddings of the transformer model following the soft token-mapping procedure described in [section 3](#), using the RobBERT tokenizer as target. Secondly, we finetune the newly-initialized embeddings and the language modeling head on a corpus of the target language. Finally, we unfreeze all the weights and continue finetuning on the same corpus.

We stop all experiments early, after using only about 15% of the available Dutch data (~7Gb text) of the Oscar19 corpus, because the training loss stabilizes around that point; another experiment using 25% of the available Dutch data, not reported in the table, showed no further improvement.

Our results, summarized in [Table 3](#), seem to indicate that models initialized with high-resources languages perform better and train faster than state-of-the-art language models, with our best model outperforming all alternatives in 4 out of the 5 tasks, while being competitive on the remaining one.

Our ablation study confirms that while finetuning the embeddings and the transformer weights on a Dutch corpus improves the performance on the downstream tasks measurably, this is not required for achieving state-of-the-art performance in most of the tasks considered in our benchmark, expanding on our observation that our initialization performs well as-is for low-resource languages. This is very promising, as it is difficult to finetune the transformer weights for many low-resource languages due to lack of available corpora (see § 4.1).

Unlike what is reported in [de Vries et al. \(2021\)](#), we find little evidence that high language similarity is critical for downstream task performance, with our Romance-initialized model (camembert-base) and our German-initialized model (gbert-base) performing about equally well on average. One notable exception to this observation concerns part-of-speech tagging, a fine-grained grammatical task; interestingly, this is the task on which [de Vries et al.](#) evaluated their converted Frisian model, which probably explains their conclusion.

## 5 Impact and Discussion

We believe the results presented above could have a large impact on the NLP community, by making the creation of high-quality models for low-resource languages more accessible. While this work focused on general-purpose models, the conversion of more specialized models, such as biomedical models, could be envisioned using specialized translation dictionaries ([Remy et al., 2022](#)).

Another interesting aspect of our approach is its potential for model distillation ([Hinton et al., 2015](#); [Buciluă et al., 2006](#)). In our second experiment, several high-resource language models were converted to Dutch models, using the same tokenizer. The usage of different tokenizers is one of the key reasons why language models are usually difficult to combine with each other, an issue that our technique can contribute to overcome.

Finally, because converting models using our technique is inexpensive, researchers using our methodology might be able to update language models on a more regular basis, an idea which the Online Language Modeling community contributed to popularize ([Thrush and Oblokulov, 2022](#)).

Our code and models will be released upon acceptance. We provide an easy-to-use notebook to convert and finetune language models for any of the 1782 language pairs supported by OpenSubtitles.

## 6 Conclusion

In this work, we were able to successfully improve the performance of language model conversion by introducing a new token translation task.

We demonstrated how this new initialization strategy largely benefits low-resource languages such as Frisian, but also makes it possible to train monolingual models for languages with more resources (such as Dutch) using far less training data and time as was previously possible.

Our results show that this approach might improve the state-of-the-art tools available for many languages across the globe, a key fairness issue.

We also believe our work is opening the gate to more frequent incremental updates of these models to keep up with the changing patterns of language over time, an often overlooked problem.

## 7 Limitations and Future Work

While this project features exciting development for low- and mid-resource languages, a couple of limitations of our work are worth discussing.

One of these limitations is that our experiments only cover Romance and Germanic languages. Collaborations to work on a more diverse set of languages are being discussed, but are at early stages.

Another limitation is that converting models to other languages might have unexpected effects on the model’s linguistic and cultural understanding. This might for instance exacerbate or dampen the biases present in the training data.

The impact of the use of a particular word translation dictionary, or the combination from multiple dictionaries, was not studied in this work, but might be important to consider as well. Using word translation dictionaries derived from aligned subtitles, like we do for Dutch, might possibly bias the language model understanding towards certain topics.

Finally, because our work relies on a token-level translation task, the understanding of multiword expressions in the target language is also a subject of concern. This understanding will probably have to be learned through the full finetuning of the model, as token embeddings are unlikely to be sufficient to learn them all in isolation. Even when the translation dictionary contains some multiword expressions, we do not provide a way to use them effectively in our proposed framework. Future works might want to provide an extension of our strategy to multiword expressions.

## References

- 800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849
- Mikel Artetxe, Vedanuj Goswami, Shruti Bhosale, Angela Fan, and Luke Zettlemoyer. 2023. [Revisiting machine translation for cross-lingual classification](#).
- Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2020a. [Translation artifacts in cross-lingual transfer learning](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7674–7684, Online. Association for Computational Linguistics.
- Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020b. [On the cross-lingual transferability of monolingual representations](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4623–4637, Online. Association for Computational Linguistics.
- Gosse Bouma and Gertjan van Noord. 2017. [Increasing return on annotation investment: The automatic construction of a Universal Dependency treebank for Dutch](#). In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 19–26, Gothenburg, Sweden. Association for Computational Linguistics.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541.
- Branden Chan, Stefan Schweter, and Timo Möller. 2020. [German’s next language model](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6788–6796, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Pieter Delobelle, Thomas Winters, and Bettina Berendt. 2020. [RobBERT: a Dutch RoBERTa-based Language Model](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3255–3265, Online. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- P. Duijff, F.J. van der Kuip, R. de Haan, and H. Sijens, editors. 2008. *Frysk Hânwurdboek*, volume 1029 of *Fryske Akademy*. Afûk / Fryske Akademy. Reporting year: 2008.
- Xavier Garcia, Noah Constant, Ankur Parikh, and Orhan Firat. 2021. [Towards continual learning for multilingual machine translation via vocabulary substitution](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1184–1192, Online. Association for Computational Linguistics.
- Evangelia Gogoulou, Ariel Ekgren, Tim Isbister, and Magnus Sahlgren. 2022a. [Cross-lingual transfer of monolingual models](#).
- 850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899

- 900 Evangelia Gogoulou, Ariel Ekgren, Tim Isbister, and Magnus Sahlgren. 2022b. [Cross-lingual transfer of monolingual models](#). In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 948–955, Marseille, France. European Language Resources Association. 950
- 901 951
- 902 952
- 903 953
- 904 954
- 905 955
- 906 956
- 907 957
- 908 958
- 909 959
- 910 960
- 911 961
- 912 962
- 913 963
- 914 964
- 915 965
- 916 966
- 917 967
- 918 968
- 919 969
- 920 970
- 921 971
- 922 972
- 923 973
- 924 974
- 925 975
- 926 976
- 927 977
- 928 978
- 929 979
- 930 980
- 931 981
- 932 982
- 933 983
- 934 984
- 935 985
- 936 986
- 937 987
- 938 988
- 939 989
- 940 990
- 941 991
- 942 992
- 943 993
- 944 994
- 945 995
- 946 996
- 947 997
- 948 998
- 949 999
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. [Distilling the knowledge in a neural network](#). *arXiv:1503.02531*.
- Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. 2018. [OpenSubtitles2018: Statistical rescoring of sentence alignments in large, noisy parallel corpora](#). In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *arXiv preprint arXiv:1907.11692*.
- Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamel Seddah, and Benoît Sagot. 2020. [CamemBERT: a tasty French language model](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.
- Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhresch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.
- Vít Novotný, Eniafé Festus Ayetiran, Dalibor Bačovský, Dávid Lupták, Michal Štefánik, and Petr Sojka. 2021. [One size does not fit all: Finding the optimal subword sizes for FastText models across languages](#). In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 1068–1074, Held Online. INCOMA Ltd.
- Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. [How multilingual is multilingual BERT?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy. Association for Computational Linguistics.
- Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. [BPE-dropout: Simple and effective subword regularization](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.
- François Remy, Peter De Jaeger, and Kris Demuynck. 2022. [Taming large lexicons : translating clinical text using medical ontologies and sentence templates](#). In *EmP : 1st RADar conference on Engineer meets Physician, Proceedings*, page 5.
- Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. [A primer in BERTology: What we know about how BERT works](#). *Transactions of the Association for Computational Linguistics*, 8:842–866.
- Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff. 2019. Masked language model scoring. *arXiv preprint arXiv:1910.14659*.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 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)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Tristan Thrush and Muhtasham Muhtasham Oblokulov. 2022. [Online language modelling training pipeline](#). HuggingFace.
- Benjamin van der Burgh and Suzan Verberne. 2019. [The merits of Universal Language Model Fine-tuning for Small Datasets – a case with Dutch book reviews](#). *arXiv:1910.00896 [cs]*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

- 1000 David Vilar and Marcello Federico. 2021. [A sta-](#) 1050  
 1001 [tistical extension of byte-pair encoding](#). In *Pro-* 1051  
 1002 *ceedings of the 18th International Conference* 1052  
 1003 *on Spoken Language Translation (IWSLT 2021)*, 1053  
 1004 pages 263–275, Bangkok, Thailand (online). As- 1054  
 1005 sociation for Computational Linguistics. 1055
- 1006 1056  
 1007 Wietse de Vries, Martijn Bartelds, Malvina Nissim, 1057  
 1008 and Martijn Wieling. 2021. [Adapting monolin-](#) 1058  
 1009 [gual models: Data can be scarce when language](#) 1059  
 1010 [similarity is high](#). In *Findings of the Association* 1060  
 1011 *for Computational Linguistics: ACL-IJCNLP* 1061  
 1012 *2021*, pages 4901–4907, Online. Association for 1062  
 1013 Computational Linguistics. 1063
- 1014 Wietse de Vries, Andreas van Cranenburgh, Ari- 1064  
 1015 ianna Bisazza, Tommaso Caselli, Gertjan van 1065  
 1016 Noord, and Malvina Nissim. 2019. BERTje: 1066  
 1017 A Dutch BERT model. *arXiv preprint* 1067  
 1018 *arXiv:1912.09582*. 1068
- 1019 Wietse de Vries and Malvina Nissim. 2021. [As](#) 1069  
 1020 [good as new. how to successfully recycle En-](#) 1070  
 1021 [glish GPT-2 to make models for other languages](#). 1071  
 1022 In *Findings of the Association for Computational* 1072  
 1023 *Linguistics: ACL-IJCNLP 2021*, pages 836–846, 1073  
 1024 Online. Association for Computational Linguis- 1074  
 1025 tics. 1075
- 1026 1076  
 1027 Zirui Wang, Zachary C. Lipton, and Yulia Tsvetkov. 1077  
 1028 2020. [On negative interference in multilingual](#) 1078  
 1029 [models: Findings and a meta-learning treatment](#). 1079  
 1030 In *Proceedings of the 2020 Conference on Em-* 1080  
 1031 *pirical Methods in Natural Language Processing* 1081  
 1032 *(EMNLP)*, pages 4438–4450, Online. Associa- 1082  
 1033 tion for Computational Linguistics. 1083
- 1034 1084  
 1035 Gijs Wijnholds and Michael Moortgat. 2021. 1085  
 1036 SICK-NL: A dataset for Dutch natural language 1086  
 1037 inference. In *Proceedings of the 16th Confer-* 1087  
 1038 *ence of the European Chapter of the Association* 1088  
 1039 *for Computational Linguistics: Main Volume*, 1089  
 1040 pages 1474–1479. 1090
- 1041 1091  
 1042 Yonghui Wu, Mike Schuster, Zhifeng Chen, 1092  
 1043 Quoc V Le, Mohammad Norouzi, Wolfgang 1093  
 1044 Macherey, Maxim Krikun, Yuan Cao, Qin Gao, 1094  
 1045 Klaus Macherey, et al. 2016. Google’s neural 1095  
 1046 machine translation system: Bridging the gap 1096  
 1047 between human and machine translation. *arXiv* 1097  
 1048 *preprint arXiv:1609.08144*. 1098
- 1049 1099  
 1047 J. W. Zantema, editor. 1984. *Frysk wurdboek. I*  
 1048 *: Frysk-Nederlânsk*, volume 1029. Drachten  
 1049 Osinga.

## A Model conversion details

In this appendix, we provide details necessary to replicate our Dutch model finetuning, as the hyperparameters required for this task are numerous, and not always relevant to mention in the main text.

Our finetuning was divided into three phases: [1] an embedding finetuning, focused on improving the initialization outcome, [2] a grammatical finetuning, focused on learning and unlearning linguistic patterns in the edge layers of the Transformer, and [3] a knowledge finetuning, where all the weights of the Transformer are finetuned.

As the number of tunable parameters varies, so must the learning rate and warmup period, to ensure high-quality results.

### A.1 Embedding finetuning

During this phase, around 5% of the Dutch corpus was used for training. In this phase, only the embedding weights and the language modeling head are tunable parameters.

```
per_device_train_batch_size=4
gradient_accumulation_steps=8,
#total_batch_size=32,
training_steps=150000,
learning_rate=5e-5,
warmup_steps=5000,
weight_decay=0.01,
fp16=True,
```

### A.2 Grammatical finetuning

In the second step, around 5% of the Dutch corpus was used for training. The main change over the previous phase are the unfreezing of the bottom two and top two Transformer layers, and the increase of the batch size for less noisy gradients.

```
per_device_train_batch_size=4
gradient_accumulation_steps=64,
#total_batch_size=256,
training_steps=25000,
learning_rate=5e-5,
warmup_steps=1000,
weight_decay=0.01,
fp16=True,
```

### A.3 Knowledge finetuning

In the second step, around 5% of the Dutch corpus was used for training. The main change over the previous phase are the unfreezing of all remaining Transformer layers, and the decrease of the learning rate to adjust to the increase of non-linear effects in the parameter updates.

```
per_device_train_batch_size=4
gradient_accumulation_steps=64,
#total_batch_size=256,
training_steps=25000,
learning_rate=2e-5,
warmup_steps=2000,
weight_decay=0.01,
fp16=True,
```

### A.4 Summary

In total, about 15% of the training data were used during the full finetuning procedure. This amounts to about 7Gb of text (out of the 47Gb available in the Oscar corpus).

We performed the tuning on a single V100 GPU, with a total running time of about a week for each model trained.

## 1200 B Dutch evaluation details 1250

1201 In this section, we provide more details about the 1251  
 1202 Dutch evaluation performed in section 4.2. 1252  
 1203

### 1204 B.1 Sentiment Analysis (SA) 1254

1205 We evaluate sentiment analysis on the Dutch Book 1255  
 1206 Review Dataset (van der Burgh and Verberne, 1256  
 1207 2019) with standard splits. This dataset is pub- 1257  
 1208 licly available with a cc-by-nc-sa-4.0 licence. Our 1258  
 1209 experiment consists of one run with the following 1259  
 1210 hyperparameters: 1260

- 1211 • Number of gpus: 1 (1080 Ti) 1261
- 1212 • adafactor: False 1262
- 1213 • adam beta1: 0.9 1263
- 1214 • adam beta2: 0.999 1264
- 1215 • adam epsilon: 1e-08 1265
- 1216 • deepspeed: None 1266
- 1217 • fp16: False 1267
- 1218 • gradient acc. steps: 8 1268
- 1219 • lr:  $10^{-4}$  1269
- 1220 • lr scheduler type: LINEAR 1270
- 1221 • num train epochs: 10 1271
- 1222 • optimizer: ADAMW 1272
- 1223 • batch size: 8 1273
- 1224 • warmup ratio: 0.0 1274
- 1225 • warmup steps: 20 1275
- 1226 • weight decay: 0.05 1276

### 1235 B.2 Named Entity Recognition (NER) 1285

1236 We evaluate NER on the CoNLL-2002 shared task 1286  
 1237 from [https://www.clips.uantwerpen.](https://www.clips.uantwerpen.be/conll2002/ner/) 1287  
 1238 [be/conll2002/ner/](https://www.clips.uantwerpen.be/conll2002/ner/) (no explicit mention 1288  
 1239 of a licence) with an experiment that consists 1289  
 1240 of 10 runs with Bayesian optimisation (TPE) 1290  
 1241 with the following hyperparameters, where 1291  
 1242 we vary the learning rate, number of gradient 1292  
 1243 accumulation steps and weight decay. We select 1293  
 1244 the best-performing model based on the  $F_1$  score 1294  
 1245 on a separate validation set before testing this 1295  
 1246 model the test set. 1296

- 1247 • Number of gpus: 1 (1080 Ti) 1297
- 1248 • adafactor: False 1298

- adam beta1: 0.9 1250
- adam beta2: 0.999 1251
- adam epsilon: 1e-08 1252
- deepspeed: None 1254
- fp16: False 1255
- gradient acc. steps: {1, 2, 4, 8, 16, 32}. 1257
- lr:  $[10^{-6}, 10^{-4}]$ . 1258
- lr scheduler type: LINEAR 1260
- num train epochs: 10 1261
- optimizer: ADAMW 1262
- batch size: 8 1264
- warmup ratio: 0.0 1265
- warmup steps: 20 1267
- weight decay: [0.01, 0.1]. 1268

### 1270 B.3 Part-of-speech (POS) tagging 1270

1271 We used the Dutch part of the Lassy cor- 1271  
 1272 pus (Bouma and van Noord, 2017) available 1272  
 1273 at [https://universaldependencies.](https://universaldependencies.org/treebanks/nl_lassysmall/index.html) 1273  
 1274 [org/treebanks/nl\\_lassysmall/index.](https://universaldependencies.org/treebanks/nl_lassysmall/index.html) 1274  
 1275 [html](https://universaldependencies.org/treebanks/nl_lassysmall/index.html) which has a cc-by-sa 4.0 licence. We 1275  
 1276 perform 10 runs with Bayesian optimisation 1276  
 1277 (TPE) with the following hyperparameters, where 1277  
 1278 we vary the learning rate, number of gradient 1278  
 1279 accumulation steps and weight decay. We select 1279  
 1280 the best-performing model based on the  $F_1$  score 1280  
 1281 on a separate validation set before testing this 1281  
 1282 model the test set. 1282

- Number of gpus: 1 (1080 Ti) 1283
- adafactor: False 1284
- adam beta1: 0.9 1285
- adam beta2: 0.999 1286
- adam epsilon: 1e-08 1287
- deepspeed: None 1288
- fp16: False 1289
- gradient acc. steps: {1, 2, 4, 8, 16, 32}. 1290
- lr:  $[10^{-6}, 10^{-4}]$ . 1291
- lr scheduler type: LINEAR 1292
- num train epochs: 10 1293
- optimizer: ADAMW 1294

- batch size: 8
- warmup ratio: 0.0
- warmup steps: 20
- weight decay: [0.01, 0.1].

#### B.4 Natural Language Inference (NLI)

We use the SICK-NL dataset (Wijnholds and Moortgat, 2021), available at [https://github.com/gijswijnholds/sick\\_nl](https://github.com/gijswijnholds/sick_nl) under the MIT licence. Our experiment consists of 30 runs with the following hyperparameters, where most are fixed and the learning rate, weight decay and the number of gradient accumulation steps are randomly selected from the specified ranges.

- Number of gpus: 1 (1080 Ti)
- adafactor: False
- adam beta1: 0.9
- adam beta2: 0.999
- adam epsilon: 1e-08
- deepspeed: None
- fp16: False
- gradient acc. steps: {2, 4, 8, 16}.
- lr:  $[10^{-6}, 10^{-4}]$ .
- lr scheduler type: LINEAR
- num train epochs: 10
- optimizer: ADAMW
- batch size: 8
- warmup ratio: 0.0
- warmup steps: 20
- weight decay: [0, 0.1].