How to do a vocab swap? A study of embedding replacement for pre-trained transformers

Anonymous authors
Paper under double-blind review

Abstract

There are a wide range of different tokenizers and vocabularies that have been used to train language models, and training a language model on just one of these can be prohibitively expensive. The ability to swap the vocabulary of a model after it has been trained enables models to be adapted to different tokenizers, and even different languages, without the computational or data cost of from-scratch training. In this paper, we ask when such swaps are possible, and how to perform them effectively? The major challenge of performing a vocab swap is re-learning the parameters of the embedding layer for the vocabulary. We observe that it is possible to re-learn the embedding for a vocabulary using a naive initialization, and we investigate strong initialization strategies that enable learning of new embeddings for swapped vocabularies, even when those vocabularies come from a different source language than the original language model.

1 Introduction

Every language modeling task begins with a tokenizer that breaks text down into a sequence of words from a fixed and finite vocabulary. The choice of tokenizer, and the resulting vocabulary of tokens it induces, is often treated as arbitrary when designing a language model. Yet, the model's vocabulary can have a major impact on the suitability of a model for downstream tasks. For example, a model trained on a standard text corpus may lack the vocabulary needed to efficiently represent common words in a set of technical documents, may lack words that have gained common usage after the model was trained, or may lack the vocabulary needed to represent words in a different (human or computer) language. This problem is compounded by the fact that performant language models are prohibitively expensive to train for many researchers and practitioners. As a result, we are usually forced to adopt the vocabulary of the language model we have rather than using a vocabulary that is ideal for the task at hand. This problem is particularly painful when working with under-resourced languages, in which case training a language model from scratch is infeasible even with unlimited/extreme computational resources.

A huge body of work has focused on adapting language models to downstream tasks, often by swapping out the last layer parameters. By contrast, less is understood about swapping a vocabulary, which requires re-training the embedding layer of a model to form a new monolingual model. Artetxe et al. (2020) and Tran (2020) made progress in their work on creating multilingual models from a monolingual models and Pfeiffer et al. (2021) tackled the problem through vocabulary extension. Additionally, Aji et al. (2020) analyzed similar problems in neural machine translation. Nevertheless, a more mechanistic understanding of how different initialization schemes affect the transferabilty of a monolingual model to a new language remains the least explored of these questions. Moreover, most common techniques often handle this problem by brute-force training a multi-language model with a very large vocabulary that encompasses the entire range of possible downstream tokens (Devlin et al., 2019; Conneau et al., 2020; Feng et al., 2020). However, it has been found that training on these large vocabularies can result in deteriorated performance, a phenomenon that has been dubbed *the curse of multilinguality* (Conneau et al., 2020).

To better understand what is required for a successful swap, we start by investigating the simpler task of throwing away the embedding layer and recovering it by training with the downstream layers frozen. Interestingly, we find that it is indeed possible to recover the unknown embedding, provided we anchor the process by initializing a small subset of tokens to their true original value. We then

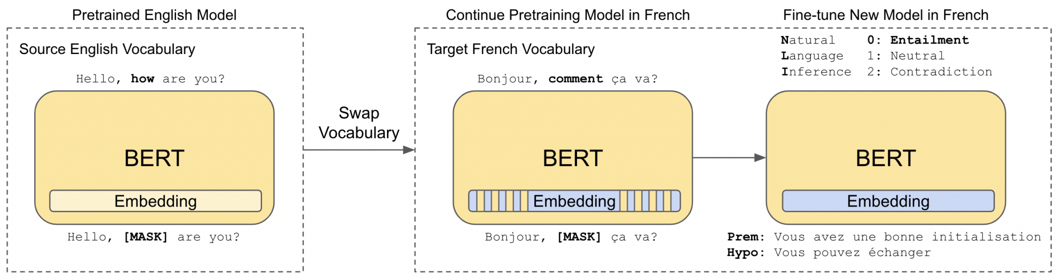


Figure 1: Swapping a vocabulary. Given a base model, BERT, we investigate strategies to swap into another vocabulary, here English to French and analyze the effects of various strategies to jump-start the new embedding. We find that a good initialization that anchors even a few key embedding correspondences can substantially improve the quality of the swap.

investigate the possibility of generating an embedding layer for a vocabulary derived from a different tokenizer, and even a completely different underlying language, than the original. Surprisingly, our observation of anchoring carries over to this much harder learning problem – anchoring a small subset of tokens with a smart initialization strategy is enough to enable fast adaptation of the model to a new language. Our main contributions are as follows: **(1)** First we introduce a simple but effective dictionary mapping that enables cross lingual transfer of monolingual model to a new language. **(2)** Then through the lens of BERT embedding recovery, a tokenizer swap, and cross lingual transfer, we compare how different initialization strategies lead to different downstream performance and geometric representations observing that anchoring is a key reason for a success swap. **(3)** Finally, we show that two different techniques, namely word alignment and dictionary matching, lead to similar representations in the word embedding matrix due to anchoring effects.

2 Problem Statement and Methodology

We center our study of the vocabulary swapping problem on the effects that different training procedures have on the token embeddings the language model learns. To this end, we use the most popular model for transfer learning and finetuning, `bert-base-uncased`, as the starting point for these experiments (Devlin et al., 2019). We use the term *source*, or *source model* to refer to such a model that is available off-the-shelf (fully pretrained) and *target* or *target vocabulary* to refer to the language and tokenization to which we would like to adapt the source model. We consider a standard pretraining and finetuning setup where we pretrain an encoder-only transformer on the masked language modeling objective (MLM), and then attach a task specific head and fine-tune the model on a downstream language understanding classification task. In the rest of the work we will regularly use the term "pretraining" to mean MLM training on the original source vocabulary, and "continued" MLM training to denote training/adapting a model on a new target vocabulary after it has been pre-trained on the source.

2.1 Initialization Schemes

We reduce the goal of producing a new monolingual model via a vocabulary swap to an initialization problem for a given model architecture and a target vocabulary. There are various options to consider for how the parameters of the target model will be initialized. In all scenarios we primarily focus on the reinitialization of the matrix of word embeddings, as the embedding matrix of the source model is indexed by the vocabulary of the source model and therefore must be modified or replaced before the model can be used in the target language. We investigate a naive baseline of random initialization and three stronger initialization methods. These strong methods correspond to various *matching schemes*, which map tokens in the source vocabulary onto tokens in the target vocabulary. This mapping can be 1-to-1 or many-to-1, and enables the target embedding matrix to be initialized using vectors from the source embedding matrix. The matching schemes we consider are described below.

From Scratch: The most trivial approach for adapting a model to a target vocabulary is to throw up your hands and train an entirely new model completely *from scratch*. This means invoking a standard weight initialization scheme for all layers in the network, and additionally replacing the embedding matrix with a new one indexed by the tokens of the target vocabulary.

Reinitialized Embedding + Pretrained Backbone (Reinit Embed): This second setting exploits the downstream language logic already learned by the source model by using its pretrained transformer backbone and only randomly initializing the embedding matrix. This methods assumes that the function learned by the transformer blocks is general purpose and can be transferred to the target modelling problem (Artetxe et al., 2020; de Vries et al., 2021). We discuss this setting in more detail in the next section.

Frequency Matching (Freq Match): The first strong initialization scheme we consider is based on the hypothesis that the semantics of the word vectors learned in the embedding matrix are intrinsically linked to their frequency distribution in the training corpus. Gogoulou et al. (2022) develop a matching scheme between two WordPiece-based tokenizers where they leverage the *implicit* frequency ordering of the output from the WordPiece algorithm. The resulting matching scheme is to initialize W_{src} using vectors from the frequency-ordered embedding W_{tgt} as $w_{tgt}^{[i]} \leftarrow w_{src}^{[i]}$. We modify this slightly and adopt what we call *explicit* frequency matching, where occurrence counts for each word in the vocabulary are independently computed over the training set in each of the respective languages (not relying on the WordPiece frequency ordering), the two vocabularies are sorted (re-indexed) by these frequency counts, $j \in [0 \to V-1]$ where $W_{tgt} \in \mathbb{R}^{(V \times d)}$, and then the initialization scheme is performed as stated above under this new index $w_{tgt}^{[j]} \leftarrow w_{src}^{[j]}$.

Alignment Matching (Align Match): The second of the stronger initialization schemes we consider is the two-stage technique described in Tran (2020) for aligning two sets of independently trained embedding vectors from different vocabularies Lample et al. (2018); Joulin et al. (2018). The first stage of the method uses fastText Bojanowski et al. (2017) as a source of pretrained word vectors in the source and target natural languages. These word vectors are aligned using the method from Joulin et al. (2018), and then each vector in the target language is assigned a group of nearest neighbord from the source language. During the second stage, the entries of W_{tgt} are initialized as linear combinations of the vectors in W_{src} as $w_{tgt}^{[i]} \leftarrow \sum_j \alpha_j w_{src}^{[j]}, \sum_j \alpha_j = 1$, where the mixing parameters $\{\alpha_j\}$ are largest for the closest neighbors. [1]

Dictionary Matching (Dict Match): Finally, we consider a simple yet surprisingly effective strategy based on access to a dictionary between the source and target vocabularies. In this scheme we initialize vectors in the target embedding using their single-word dictionary translation if one exists. We can build this dictionary on demand for any pair of source and target vocabularies using Google Translate, assuming that the source and target are natural languages. This provides a simple and accurate matching for many commonly used words.

2.2 Backbone Parameters and Model Gradients

In additional to investigating the role of the initialization schemes described above, we also consider the following training settings.

Transformer Backbone Components: While we primarily focus our analysis of initialization schemes and their effects on the word embedding matrix, we also study how the transfer of all the other parameters of the transformer – positional embeddings, attention projections, feed-forward layer weights, and layer norm parameters – affects the vocabulary swap process. In select experiments we randomly initialize the word embedding matrix only, and transfer the pretrained positional embeddings from the source model, but in others, all parameters before and after the transformer blocks are randomly initialized as done in (Artetxe et al., 2020).

Parameter Freezing: We consider two primary policies for gradient flow during continued MLM training and downstream finetuning. The default setting is to allow all of the model parameters to be updated during training by computing all gradients, denoted **All Grad**. The other is **Embed Grad**, where we only update the parameters in the embedding layer (the backbone is frozen). In a few isolated cases in service of controlled analysis, we test even further restricting gradient flow

[1]We note that this is within the scope of our problem formulation because these word vectors are learned without parallel corpora, and without needing to train an expensive transformer encoder with them, so like the source transformer parameters, we consider this to be available "off-the-shelf".

to just the word embedding matrix, freezing the positional embeddings as well as the layer norm parameters. We clearly differentiate this variant where relevant.

2.3 Lenses of Analysis

Our empirical analyses is motivated by a few hypotheses. First, we suspect that learning an appropriate embedding for the new vocabulary, rather than updating the backbone transformer, is the primary challenge during a vocabulary swap as it represents a significant domain shift from the perspective of the downstream layers. Second, two high-performance models that share a vocabulary and downstream backbone should also share similar embedding matrices. To investigate these claims, we will run experiments where we learn the embedding layer of a model twice in two different ways, and then compare the embeddings learned. We consider the following measure of similarity.

Embedding Correlation: *The correlation between two word embedding matrices W_1 and W_2, both indexed by the token ids of the same vocabulary V, is defined as the average pearson correlation of corresponding word vectors from the two embeddings:*

$$\bar{\rho}(W_1, W_2) = \frac{1}{|V|}\sum \rho_i, \quad \rho_i = \frac{COV(x_i, y_i)}{\sigma_{x_i}\sigma_{y_i}}, \quad \forall i \in V$$

Model Performance: We ground our embedding analysis to the real cross lingual transfer setting by quantifying the effectiveness of various initialization and training strategies using standard performance metrics. For continued MLM pretraining in a target language, we measure the negative log likelihood on a validation set and for finetuning tasks we measure the standard evaluation metric for the classification dataset, generally accuracy on the validation set.

3 Experimental Setup

We consider three classes of experiments, two synthetic swap scenarios that give us a lot of control over the vocabularies and embeddings, and a traditional cross-lingual transfer experiment.

3.1 Experiment 1: BERT Embedding Recovery

The value of a case study in recovering BERT embeddings derives from the controlled nature of the experimental setting. We suspect that the success of a cross lingual transfer under the traditional language swap scenario will be predicated on the feasibility of simply relearning a specific known embedding, under various levels of perturbation and different kinds of reinitialization. Starting with a fully pretrained BERT checkpoint, we perturb the embedding and then continue training to see how accurately the original embedding is recovered. We experiment with the following methods for perturbing the embedding with various levels of severity.

Permute: We explore completely scrambling the vectors, and also scrambling them within a frequency window of width 100. For example, for token id from 100 to 200, token id 114 is now initialized with a random word vector from that same window. We use the permutation to understand if frequency is important for initialization as postulated by (Gogoulou et al., 2021).

Keeping $x\%$ of Word Vectors: We uniformly select $x\%$ of random word vectors to keep, and randomize the rest. This is to mimic a dictionary mapping or other more complex word alignment techniques like (Tran, 2020).

We compare these initializations to the `Scratch` and `Reinit Embed` settings described in Section 2. As a reference point we also consider the performance an unmodified `bert-base-uncased` model that we continue to train – this provides a baseline with the same amount of total training as the experimental models. We study both what happens when you reinitialize the whole embedding versus just the word embedding layer to understand how positional embeddings affect the word representations learned. For these pre-training runs, we continue training on the MLM objective–using the canonical 15% masked tokens per sequence–with a sequence length of 128 for 40k steps and 512 for 10k steps to emulate (Devlin et al., 2019).[2] After MLM

[2]Initial results showed that 5k steps or 10% of the total steps of sequence length 512 was not sufficient.

training, we use MNLI Williams et al. (2018), MRPC Dolan & Brockett (2005), and STSB Cer et al. (2017) from the GLUE benchmark to measure the effect that the different initialization schemes for MLM training have on downstream classification performance.

3.2 Experiment 2: FrankenBERTa

In this experiment we start with a pretrained `roberta-base` model, which uses its native Byte-Pair-Encoding (BPE) vocabulary, and attempt to swap it to use a `bert-base-uncased` tokenizer, whose vocabulary is generated using the WordPiece algorithm. Our hope is that the more powerful RoBERTa model, after a vocab swap, can out-compete BERT on its home turf. We run experiments using the Scratch, Reinit Embed, Freq Match, and Dict Match initialization schemes and we train in the All-Grad and Embed-Grad settings using the same pre-training set-up and downstream tasks as in the BERT Recovery experiments. [3]

3.3 Experiment 3: Cross-Lingual Transfer (CLT)

Finally, to study the realistic scenario of retrofitting a pretrained model to process a completely different underlying natural language, we perform CLT experiments in which an English `bert-base-uncased` model is vocab-swapped to accept the WordPiece vocabulary from the `dbmdz/bert-base-german-cased` model. We also consider a swap to the BPE tokenizer vocabulary for the French `flaubert/flaubert_base_cased` model (Le et al., 2019). In these experiments we consider GermanBERT and FlauBERT as the optimal reference models. We run experiments across our battery of initialization schemes and two gradient control settings.

Shared Implementation Details: For all experimental settings except FrankenBERTa, we use a `bert-base-uncased` model, which has an embedding layer consisting of word vectors, positional encodings, and token type sub components, the outputs of which are added together before applying a layer norm and passing the outputs to the first self-attention block. For all English settings, the dataset is a 50% slice of the `bookcorpus` and `wikipedia` datasets. We use an effective batch size of 256 when training with sequence length 128, and 48 effective batch size when training sequence length 512. For the CLT pre-training runs, we continue training on the MLM objective with a sequence length of 128 for 90k steps and 512 for 10k steps. For the German and French CLT settings, we use Wikipedia datasets derived from dumps of french and German Wikipedia respectively. Here we use the downstream tasks that are equivalent to MNLI and MPRC, XNLI introduced by Conneau et al. (2018) and PAWX-S introduced by Yang et al. (2019) from the XTREME dataset Hu et al. (2020), and Pearson correlation relative to other setting types. For the downstream experiments on XNLI, we train and test in the target language, using the official machine translations provided.

4 Results and Discussion

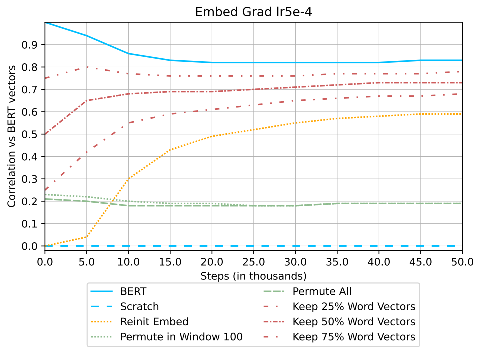
We observe that, for toy embedding recovery problems, random initialization of embeddings generally performs poorly, while stronger initializations result in an "anchoring" effect that enables fairly accurate recovery of the original embedding. Interestingly, this lesson carried over to the cross-lingual setting. There too, we see that anchored initializations facilitate effective learning of embeddings.

4.1 BERT Embedding Recovery

Figure 2 and Figure 7 in Appendix A.1 show correlations between the original and recovered embeddings when trying to recover a BERT embedding that has been corrupted by various means.

The first thing to observe is that random initialization of the embedding makes recovery quite difficult. When updating all the layers or training just the embedding with a lower learning rate, we find

[3] We omit the Align Match technique here because when swapping between different vocabularies over the same underlying language, the first step of the alignment process degenerates to a form of dictionary matching since the two sets of fastText vectors being aligned would actually be the same, with a trivial (identity) translation matrix between them. Additionally, Dict Match devolves into an exact match between the two tokenizers.



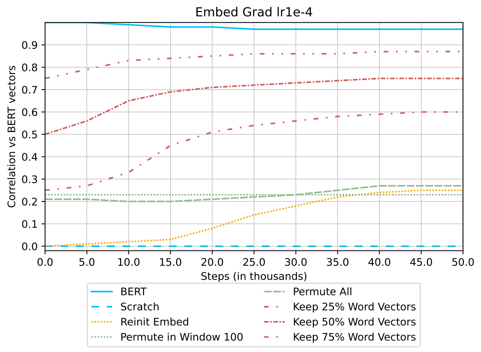


Figure 2: The correlation between the pretrained BERT-baseline embedding matrix and a retrained embedding matrix over 50k step in increments of 5k. Each figure shows the different initialization schemes: permute (whole and in window), random initialization, reinitialize word vectors (25%, 50%, 75% kept). **Left** shows retraining all embedding layers with a peak LR of 5e-4 after reinitialization all embedding layers and initializing with the given scheme. **Right** shows retraining all embedding layers after reinitialization with a peak LR of 1e-4.

that random initialization yields a poor recovery. Training the embedding with a higher learning rate improves performance, but still not to the degree possible with better initializations.

We find that anchoring the embedding matrix by *keeping* a percentage of the embedding vectors forces the model to have a higher correlation with the original `bert-base-uncased` word embedding in all settings we explored. Figure 7 shows the case where we constrain the problem so only the word embedding can be updated. In this case, *permutation* initialization does recover the embedding to a some degree of correlation. However, we see that performance of weak initializations are quite fragile, namely *permutation* and *Reinit Embed* are quite sensitive to learning rates and the choice of which layers to update.

Table 1 shows that downstream MNLI performance is mostly recovered by just training the embedding layers for all settings except for *scratch* and the *permutation* initializations. The stronger anchored initalization is quite successful, and unsurprisingly the more word vectors are kept from the original BERT embedding the better the performance. When comparing these numbers to those in Figure 2, we see that upstream MLM performance is quite predictive of MNLI performance. Furthermore, we see that Reinit Embed performance varies with the training setup, as this initialization in more sensitive to the training scheme. Overall, we see that one can recover the performance of the original model when the initialization is favorable. In Table 4 and 6, we see a similar pattern with MRPC and STSB results.

	MNLI (Acc. m/mm)	
	Embed Layers	
	1e-4	5e-4
Bert Baseline	0.842/0.846(±0.004/0.004)	0.839/0.842(±0.003/0.004)
Scratch	0.7/0.708(±0.003/0.001)	0.702/0.709(±0.004/0.007)
Reinit Embed	0.82/0.821(±0.004/0.006)	0.829/0.832(±0.003/0.003)
25% Kept	0.83/0.834(±0.007/0.007)	0.831/0.831(±0.002/0.002)
50% Kept	0.833/0.834(±0.004/0.006)	0.83/0.834(±0.004/0.003)
75% Kept	**0.836/0.837(±0.005/0.004)**	**0.833/0.837(±0.004/0.007)**
Permute	0.777/0.781(±0.025/0.022)	0.754/0.763(±0.004/0.003)
Permute 100	0.734/0.74(±0.003/0.006)	0.726/0.74(±0.051/0.052)

Table 1: MNLI average accuracy over five seeds for Embed Grad for learning 1e-4 and 5e-4 after 50k of continue training with different initialization schemes. We see that we can mostly recover the performance of BERT in most initialization schemes. Permutation initializations and Scratch do not perform well compared to other methods.

Anchoring Effect: Figure 3 shows the anchoring effect of have a subset of word vectors initialized from the original embedding. As training proceeds, the randomly initialized word vectors quickly

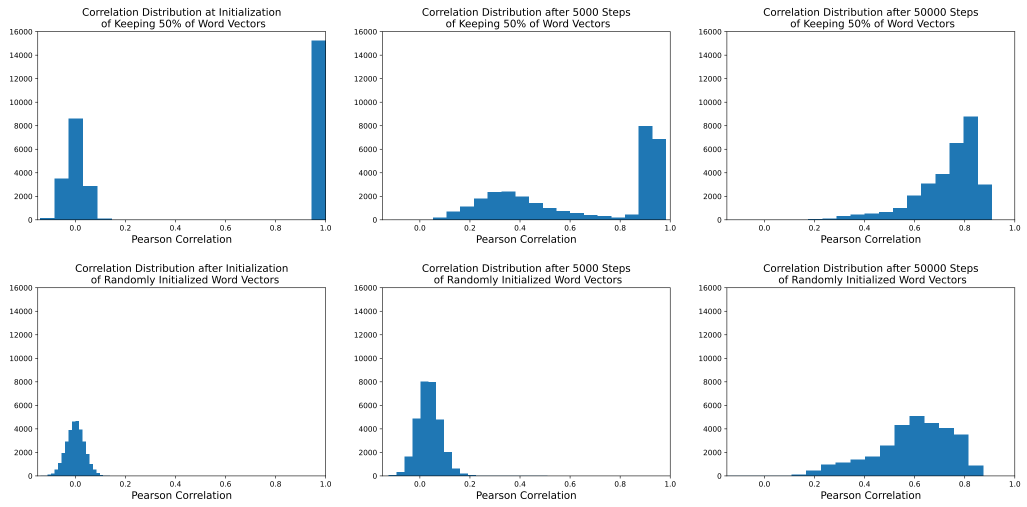


Figure 3: **Top:** The distribution of Pearson Correlation between the word vectors of BERT and embedding that is being trained, when 50% of embedding rows are kept. Shown are Initialization (**left**), step 5000 **(middle)**, and step 50000 **(right)**. **Bottom:** The distribution of Pearson Correlation between the word vectors of BERT and embedding that is being trained, random embedding initialization at Initialization (**left**), step 5000 **(middle)**, and step 50000 **(right)**. We see a distinct *anchoring effect* comparing both rows, as the randomly initialized vectors quickly move toward high correlation with the original BERT word vectors.

revert towards their original values. Essentially the kept words seem to uniquely identify which loss function basin the other vector should fall into. This effect is observed in the case of Reinit Embed, but to a much weaker degree. This anchoring effect is a powerful tool, and it will enable us to boost performance of vocab swaps in harder settings.

4.2 FrankenBERTa

		STSB (Pearson)	MRPC (F1)	MNLI (m/mm Acc.)
Dict	LR 1e-4 (All)	0.89(±0.003)	0.907(±0.006)	0.868/0.867(±0.002/0.002)
Freq	LR 1e-4 (All)	0.847(±0.01)	0.89(±0.004)	0.833/0.834(±0.002/0.003)
Reinit Emebd	LR 1e-4 (All)	0.859(±0.006)	0.895(±0.004)	0.836/0.838(±0.002/0.001)
Dict	LR 5e-4 (Embed)	**0.897(±0.004)**	0.904(±0.007)	**0.874/0.874(±0.001/0.001)**
Freq	LR 5e-4 (Embed)	0.878(±0.004)	**0.911(±0.002)**	0.867/0.866(±0.001/0.002)
Reinit Emebd	LR 5e-4 (Embed)	0.888(±0.002)	0.904(±0.007)	0.868/0.87(±0.001/0.001)
`bert-base-uncased`	Pre-trained	0.886(±0.003)	0.886(±0.011)	0.847/0.848 (±0.001/0.003)
`roberta-base`	Pre-trained	**0.901(±0.004)**	**0.915(±0.01)**	**0.879/0.877 (±0.002/0.001)**

Table 2: STSB, MRPC and MNLI average results over five seeds for RoBERTa models modified to work with the BERT tokenizer. We also report the original BERT and RoBERTa performance as a baseline. We see that we can produce a stronger WordPiece model than the original BERT in this setting. Positional embeddings were not reinitialized.

Above, we studied an anchoring effect that enables us to re-learn embeddings from strong initializations. Going forward, we want to know whether this anchoring effect is still present for harder swaps. We now try to learn a better BERT WordPiece vocabulary language model than the original BERT just by swapping the vocabulary of the more powerful RoBERTa backbone.

We find that, under the Dict Match scheme, the swapped model achieves an MLM loss of 1.52 versus the original BERT model's 1.723 shown in Figure 8 in the Appendix. Furthermore, across all three downstream tasks in Figure 10, both the Dict Match and weaker Freq Match scheme achieve the best results, outperforming BERT base, and nearly tying the performance of the original unmodified RoBERTa.

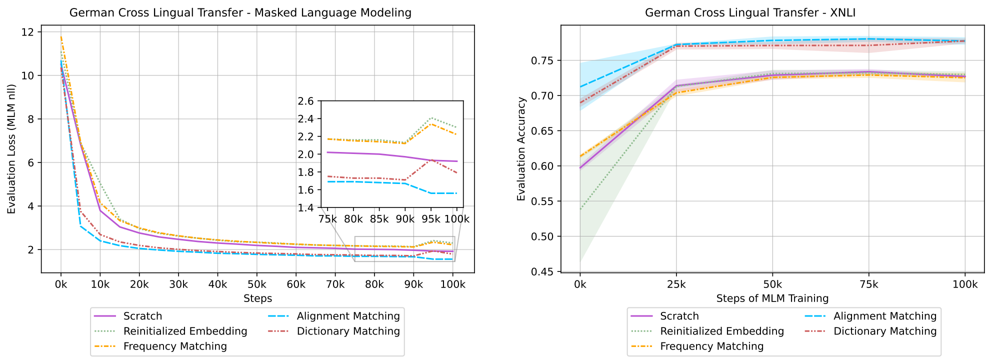


Figure 4: Performance on the validation set over the course of continued MLM training on German Wikipedia and evaluation accuracy after finetuning on German XNLI at five different points during the full MLM training run.

4.3 Full Cross Lingual Transfer

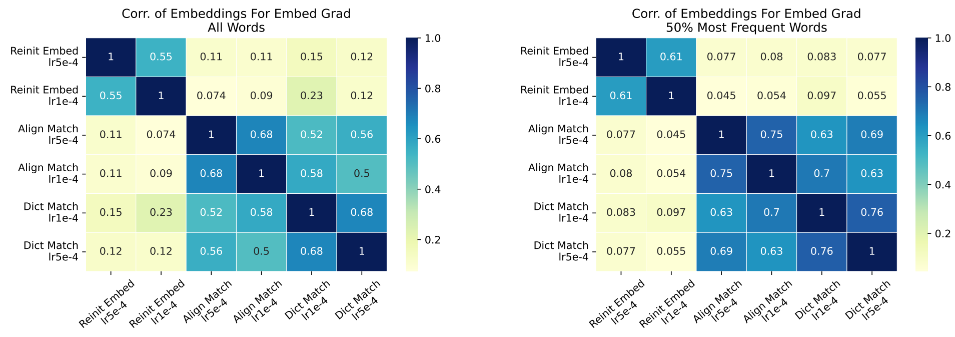
In cross lingual transfer, our goal is to re-purpose the English BERT model to understand either French or German. We found that copying information about the word vectors from the source model using either Dict Match or Align Match results in downstream classification performance that is close to that of a fully trained BERT model in that language. The results from Table 12 show that we achieve accuracy only 2% short of the FlauBERT reference model. Table 15 illustrates that a smart initialization scheme enables performance only 0.5% shy of German BERT on XNLI. Table 13 and 16 show similar results for PAWS-X. Moreover, Figure 4 shows that the recovery is quite quick when matching initializations are used. Although Dict Match and Align Match are highly competitive, Freq Match and Reinit Embed failed to out-perform end-to-end from-scratch training.

Furthermore, from Figure 5, we can see that Dict Match and Align Match have fairly high embedding correlation with each other. This correlation is even tighter in the top 50% most frequent words. This reinforces the idea of "anchoring" – both methods seem to converge to a very similar loss basin preferred by the model, despite starting from different locations. This "closeness" is also reflected in the performance realized in the PAWS-X and XNLI tasks. The Reinit Embed methods produce substantially different results from the strong initializations. Note that both Reinit runs started with the same random seed. The fact they only achieve 50% correlation despite starting in the same place is an indicator that their performance is strongly dependent on learning rate, just as we say with our toy problem.

Anchoring Effect: Figure 6 shows the token-wise accuracy (matched/not matched) for Dict Match for 1k most frequent words for the first 5k steps with histograms of the token-wise distribution for the matched tokens and not matched tokens for step 1000 and step 5000. We see evidence that the Dict Match scheme creates an anchoring effect by bootstrapping the accuracy of the non-matched tokens to well above the accuracy of those same tokens under the from scratch setting.

5 Related Work

Instead of training models from scratch for zero-shot cross-lingual transfer with large vocabularies such as XLM-R Conneau et al. (2020), Tran (2020) and Artetxe et al. (2020) propose techniques for creating multilingual models from English BERT. They first retrain word embedding for the target language, then fine-tune the whole model in the source language, and finally evaluate the zero-shot performance in the target language. Additionally, Artetxe et al. (2020) show that simply retraining the embedding layer from a random initialization is competitive with multilingual BERT on standard cross-lingual classification benchmarks suggesting that the success of multilingual models may not be predicated on joint training and a shared sub-word vocabulary as postulated by (Pires et al., 2019). Additionally, Garcia et al. (2021), Lakew et al. (2019), Lakew et al. (2018) have explored this vocabulary problem in the context of NMTs.



	Reinit Embed lr5e-4	Reinit Embed lr1e-4	Align Match lr5e-4	Align Match lr1e-4	Dict Match lr1e-4	Dict Match lr5e-4
Reinit Embed lr5e-4	1	0.55	0.11	0.11	0.15	0.12
Reinit Embed lr1e-4	0.55	1	0.074	0.09	0.23	0.12
Align Match lr5e-4	0.11	0.074	1	0.68	0.52	0.56
Align Match lr1e-4	0.11	0.09	0.68	1	0.58	0.5
Dict Match lr1e-4	0.15	0.23	0.52	0.58	1	0.68
Dict Match lr5e-4	0.12	0.12	0.56	0.5	0.68	1

	Reinit Embed lr5e-4	Reinit Embed lr1e-4	Align Match lr5e-4	Align Match lr1e-4	Dict Match lr1e-4	Dict Match lr5e-4
Reinit Embed lr5e-4	1	0.61	0.077	0.08	0.083	0.077
Reinit Embed lr1e-4	0.61	1	0.045	0.054	0.097	0.055
Align Match lr5e-4	0.077	0.045	1	0.75	0.63	0.69
Align Match lr1e-4	0.08	0.054	0.75	1	0.7	0.63
Dict Match lr1e-4	0.083	0.097	0.63	0.7	1	0.76
Dict Match lr5e-4	0.077	0.055	0.69	0.63	0.76	1

Figure 5: Embedding correlation between Reinit Embed, Align Match, and Dict Match at the end of continued pre-training in the French target vocabulary. **(Left)** shows the average correlation between all tokens of the embeddings. **(Right)** shows the correlation for the 50% most frequent tokens.

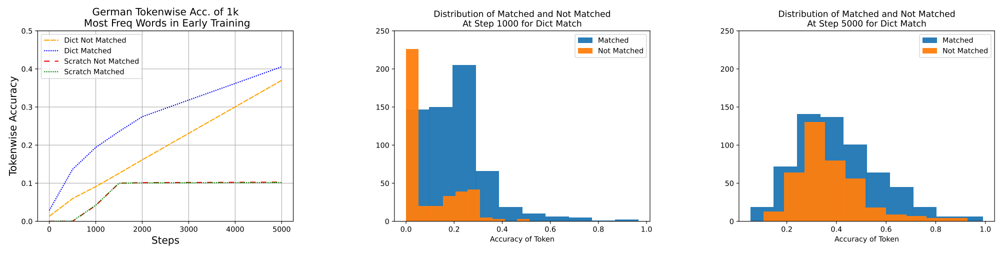


Figure 6: **(Left)** Tokenwise average accuracy for Dict Match for the top 1k words during first 5k steps. **(Middle)** shows the distribution of matched versus not matched for Dict Match at 1k steps. **(Right)** shows the distribution of matched versus not matched for Dict Match at 5k steps.

de Vries & Nissim (2020) convert a casual language model, GPT-2, from English to Italian and Dutch by retraining the embedding layer to show that transformer models can share representations across languages. Gogoulou et al. (2021) consider the setting of two models with WordPiece tokenizers and used the natural frequency ordering of their vocabularies as a scheme to initialize the embedding for the new language (called frequency matching in our study) and perform pre-training and finetuning in the target language, thereby creating a new monolingual model from an existing one. Additionally, de Vries & Nissim (2021) find performance is correlated with “distance” between the source and target languages when retraining the word embedding layer from a random initialization with limited data. Finally, Wu et al. (2022) found that continued pre-training in the target language after reinitializing the embedding improves performance.

6 Conclusion

We investigate the feasibility of vocabulary swaps for language models based on transformer architectures. We categorize a variety of initialization strategies that have the potential to warm-start such a swap. Empirically, we then first verify that it is indeed possible to “retrain the same embedding twice”, in a control study of BERT embedding recovery. We move to a swap between two English-language tokenizers, finding that with the right strategy, a BERT tokenizer can be swapped onto a roBERTa model, and result in an improved model. Finally, we analyze swaps from English into French and German where we find that clever initializations through matching can almost match the performance of models trained entirely in these languages. Throughout our analysis we study the anchoring effect of a good initialization, and observe its important in achieving a successful swap. We hope that the intuitions we develop here can contribute to the more flexible use of pre-trained models, and the development of models for a broad range of languages.

REFERENCES

Alham Fikri Aji, Nikolay Bogoychev, Kenneth Heafield, and Rico Sennrich. In neural machine translation, what does transfer learning transfer? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7701–7710, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.688. URL https://aclanthology.org/2020.acl-main.688.

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4623–4637, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.421. URL https://aclanthology.org/2020.acl-main.421.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL https://aclanthology.org/Q17-1010.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pp. 1–14, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001. URL https://aclanthology.org/S17-2001.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2475–2485, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269. URL https://aclanthology.org/D18-1269.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised Cross-lingual Representation Learning at Scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL https://www.aclweb.org/anthology/2020.acl-main.747.

Wietse de Vries and Malvina Nissim. As good as new. how to successfully recycle english gpt-2 to make models for other languages. *arXiv preprint arXiv:2012.05628*, 2020.

Wietse de Vries and Malvina Nissim. As Good as New. How to Successfully Recycle English GPT-2 to Make Models for Other Languages. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 836–846, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.74. URL https://aclanthology.org/2021.findings-acl.74.

Wietse de Vries, Martijn Bartelds, Malvina Nissim, and Martijn Wieling. Adapting monolingual models: Data can be scarce when language similarity is high. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 4901–4907, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.433. URL https://aclanthology.org/2021.findings-acl.433.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. URL https://aclanthology.org/I05-5002.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

Xavier Garcia, Noah Constant, Ankur Parikh, and Orhan Firat. 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*, pp. 1184–1192, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.93. URL `https://aclanthology.org/2021.naacl-main.93`.

Evangelia Gogoulou, Ariel Ekgren, Tim Isbister, and Magnus Sahlgren. Cross-lingual transfer of monolingual models. *arXiv preprint arXiv:2109.07348*, 2021.

Evangelia Gogoulou, Ariel Ekgren, Tim Isbister, and Magnus Sahlgren. Cross-lingual Transfer of Monolingual Models, May 2022. URL `http://arxiv.org/abs/2109.07348`. arXiv:2109.07348 [cs].

Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2979–2984, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1330. URL `https://aclanthology.org/D18-1330`.

Surafel M. Lakew, Aliia Erofeeva, Matteo Negri, Marcello Federico, and Marco Turchi. Transfer learning in multilingual neural machine translation with dynamic vocabulary. In *Proceedings of the 15th International Conference on Spoken Language Translation*, pp. 54–61, Brussels, October 29-30 2018. International Conference on Spoken Language Translation. URL `https://aclanthology.org/2018.iwslt-1.8`.

Surafel M. Lakew, Alina Karakanta, Marcello Federico, Matteo Negri, and Marco Turchi. Adapting multilingual neural machine translation to unseen languages. In *Proceedings of the 16th International Conference on Spoken Language Translation*, Hong Kong, November 2-3 2019. Association for Computational Linguistics. URL `https://aclanthology.org/2019.iwslt-1.16`.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *International Conference on Learning Representations*, 2018.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, and Didier Schwab. Flaubert: Unsupervised language model pre-training for french. *arXiv preprint arXiv:1912.05372*, 2019.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. UNKs everywhere: Adapting multilingual language models to new scripts. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10186–10203, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.800. URL `https://aclanthology.org/2021.emnlp-main.800`.

Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1493. URL `https://aclanthology.org/P19-1493`.

Ke Tran. From english to foreign languages: Transferring pre-trained language models. *arXiv preprint arXiv:2002.07306*, 2020.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL `https://aclanthology.org/N18-1101`.

Zhengxuan Wu, Isabel Papadimitriou, and Alex Tamkin. Oolong: Investigating what makes crosslingual transfer hard with controlled studies. *arXiv preprint arXiv:2202.12312*, 2022.

Yinfei Yang, Yuan Zhang, Chris Tar, and Jason Baldridge. Paws-x: A cross-lingual adversarial dataset for paraphrase identification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3687–3692, 2019.

A Appendix

A.1 Further Implementation Details

We used a linear decay learning rate with a peak of 5e-4 and 1e-4 when training the embedding layer or the word embedding layer, and used a peak learning rate of 1e-4 and 5e-5 for updating all the layers. A learning rate 5e-4 was shown to be unstable when updating all layers of the model in preliminary experiments. [4] All continued pre-training experiments were done on NVIDIA RTXA5000 GPUs. For downstream tasks, we use a single GPU and the default hyperparameters from the HuggingFace library.

Tokenizer	Algorithm	Approximate Vocab Size
`bert-base-uncased`	Wordpiece	30k
`roberta-base`	Byte-Pair Encoding	50k
`dbmdz/bert-base-german-cased`	Wordpiece	30k
`flaubert/flaubert_base_cased`	Byte-Pair Encoding	68k

Table 3: Huggingface Tokenizer Name with algorithm used and approximate vocabulary size.

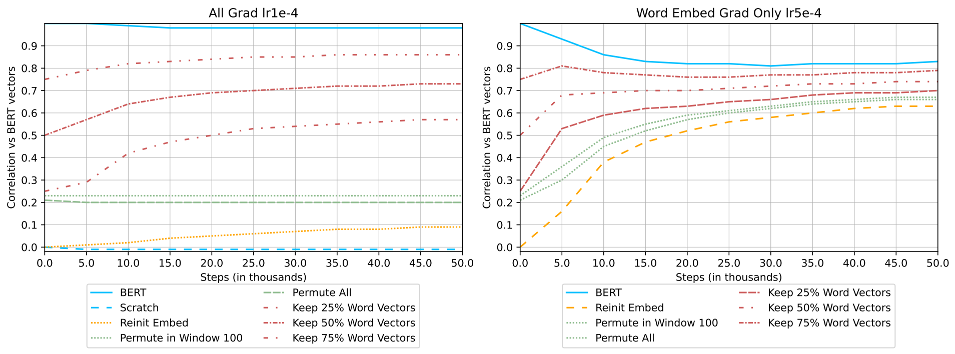


Figure 7: The correlation between the pretrained BERT-baseline embedding matrix and a retrained embedding matrix over 50k step in increments of 5k. Each figure shows the different initialization schemes: permute (whole and in window), random initialization, reinitialize word vectors (25%, 50%, 75% kept). **Left** shows retraining all layers after reinitialization with a peak LR of 1e-4. **Right** shows retraining just the word embedding layers after reinitialization of only the word embedding layer with a peak LR of 5e-4.

[4]When training only the word embedding, sequence length of 128 was used throughout training since the positional embeddings are preserved.

	MRPC (F1)			
	All Layers		Embed Layers	
	1e-4	5e-5	1e-4	5e-4
Bert Baseline	0.895(±0.006)	0.897(±0.007)	0.895(±0.005)	0.881(±0.007)
Scratch	0.833(±0.01)	0.813(±0.004)	0.809(±0.003)	0.807(±0.006)
Reinit Embed	0.859(±0.009)	0.847(±0.013)	0.861(±0.009)	0.879(±0.006)
25% Kept	0.871(±0.003)	0.871(±0.011)	0.857(±0.006)	**0.889(±0.01)**
50% Kept	0.87(±0.013)	0.877(±0.007)	**0.876(±0.008)**	0.881(±0.003)
75% Kept	**0.877(±0.007)**	**0.892(±0.003)**	0.87(±0.012)	0.86(±0.004)
Permute	0.856(±0.004)	0.864(±0.018)	0.881(±0.009)	0.849(±0.004)
Permute 100	0.86(±0.008)	0.862(±0.009)	0.846(±0.007)	0.854(±0.011)

Table 4: MRPC F1 average over five seeds of different perturbations to the BERT embedding. We see that keeping some percent of the initial vectors is the most successful strategy.

	STSB (Pearson Correlation)			
	All Layers		Embed Layers	
	1e-4	5e-5	1e-4	5e-4
Bert Baseline	0.871(±0.002)	0.875(±0.004)	0.882(±0.007)	0.862(±0.005)
Scratch	0.823(±0.009)	0.793(±0.023)	0.178(±0.008)	0.187(±0.017)
Reinit Embed	0.837(±0.006)	0.84(±0.004)	0.838(±0.005)	**0.863(±0.004)**
25% Kept	0.854(±0.002)	0.85(±0.007)	**0.855(±0.007)**	0.859(±0.003)
50% Kept	0.858(±0.001)	0.853(±0.005)	0.848(±0.014)	0.849(±0.007)
75% Kept	**0.86(±0.004)**	**0.862(±0.003)**	0.853(±0.004)	0.846(±0.006)
Permute	0.835(±0.009)	0.816(±0.007)	0.83(±0.005)	0.813(±0.008)
Permute 100	0.837(±0.003)	0.825(±0.003)	0.822(±0.003)	0.819(±0.003)

Table 5: STSB Pearson Correlation average over five seeds of different perturbations to the BERT embedding. We see that keeping some percent of the initial vectors is on average the most successful strategy.

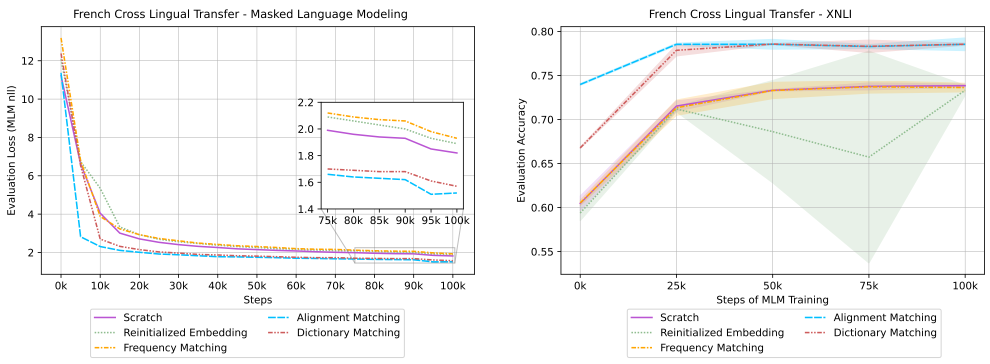


Figure 8: Performance on the validation set over the course of continued MLM training on French Wikipedia and evaluation accuracy after finetuning on French XNLI at five different points during the full MLM training run.

	MNLI (Acc. m/mm)	
	All Layers	
	1e-4	5e-5
Bert Baseline	0.838/0.842(±0.004/0.004)	0.841/0.844(±0.002/0.002)
Scratch	0.776/0.778(±0.003/0.002)	0.748/0.754(±0.002/0.002)
Reinit Embed	0.808/0.813(±0.003/0.004)	0.804/0.81(±0.003/0.006)
25% Kept	0.833/0.835(±0.002/0.002)	0.832/0.834(±0.005/0.004)
50% Kept	0.833/0.837(±0.004/0.003)	0.835/0.836(±0.002/0.003)
75% Kept	**0.835/0.837(±0.001/0.001)**	**0.837/0.838(±0.003/0.002)**
Permute	0.779/0.788(±0.005/0.003)	0.764/0.773(±0.003/0.005)
Permute 100	0.777/0.787(±0.004/0.004)	0.763/0.77(±0.002/0.003)

Table 6: MNLI accuracy average over five seeds of different perturbations to the BERT embedding. We see that keeping some percent of the initial vectors is the most successful strategy.

	MNLI (Acc. m/mm)	
	Embed Layers	
	1e-4	5e-4
Bert Baseline	0.842/0.846(±0.004/0.004)	0.839/0.842(±0.003/0.004)
Scratch	0.7/0.708(±0.003/0.001)	0.702/0.709(±0.004/0.007)
Reinit Embed	0.82/0.821(±0.004/0.006)	0.829/0.832(±0.003/0.003)
25% Kept	0.83/0.834(±0.007/0.007)	0.831/0.831(±0.002/0.002)
50% Kept	0.833/0.834(±0.004/0.006)	0.83/0.834(±0.004/0.003)
75% Kept	**0.836/0.837(±0.005/0.004)**	**0.833/0.837(±0.004/0.007)**
Permute	0.777/0.781(±0.025/0.022)	0.754/0.763(±0.004/0.003)
Permute 100	0.734/0.74(±0.003/0.006)	0.726/0.74(±0.051/0.052)

Table 7: MNLI accuracy average over five seeds of different perturbations to the BERT embedding. We see that keeping some percent of the initial vectors is the most successful strategy.

	MLM loss at 50k Steps					
	Embed Layers		All Layers		Word Embed Only	
	1e-4	5e-4	5e-5	1e-4	1e-4	5e-4
Reinit Embed	1.87	1.82	1.58	1.56	2.26	1.96
75% Kept	1.9	1.83	1.6	1.58	1.91	1.91
50% Kept	1.92	1.86	1.66	1.62	1.93	1.91
25% Kept	1.99	1.81	1.77	1.66	2.00	1.95
Permute	3.05	3.2	2.65	2.3	2.23	1.95
Permute 100	3.83	3.29	2.72	2.27	2.18	1.96
Scratch	-	-	2.97	2.39	-	-

Table 8: MLM loss of different BERT initializations and training protocols.

MLM loss RoBERTa backbone with BERT tokenizer

	Embed Grad	All Grad
	LR 5e-4	LR 1e-4
Reinit Embed	1.64	1.86
Freq Match	1.71	1.99
Dict Match	1.56	1.52

Table 9: MLM loss of different three initializations and two training protocols.

		STSB (Pearson)	MRPC (F1)	MNLI (m/mm Acc.)
Dict	LR 1e-4 (All)	**0.84(±0.003)**	0.868(±0.008)	0.827/0.831(±0.001/0.002)
Freq	LR 1e-4 (All)	0.803(±0.003)	0.87(±0.005)	0.794/0.801(±0.001/0.003)
Reinit Emebd	LR 1e-4 (All)	0.816(±0.002)	0.861(±0.005)	0.777/0.785(±0.004/0.004)
Dict	LR 5e-4 (Embed)	0.839(±0.003)	**0.888(±0.009)**	**0.829/0.835(±0.002/0.001)**
Freq	LR 5e-4 (Embed)	0.837(±0.007)	0.875(±0.011)	0.822/0.825(±0.005/0.005)
Reinit Emebd	LR 5e-4 (Embed)	0.81(±0.007)	0.88(±0.006)	0.822/0.826(±0.003/0.003)
`bert-base-uncased`	Pre-trained	0.886(±0.003)	0.886(±0.011)	0.847/0.848 (±0.001/0.003)
`roberta-base`	Pre-trained	**0.901(±0.004)**	**0.915(±0.01)**	**0.879/0.877(±0.002/0.001)**

Table 10: STSB, MRPC and MNLI average results over five seeds. Swapping the Wordpiece BERT tokenizer to the BPE RoBERTa tokenizer. We see that Dict Match gets the closest performance to BERT, but does not recover.

French MLM loss at 100k Steps

	Embed Layers		All Layers	
	1e-4	5e-4	5e-5	1e-4
Scratch	-	-	-	1.82
Embed Reinit	3.07	2.94	1.89	1.75
Freq Match	-	-	1.94	1.74
Align Match	2.31	2.42	1.52	1.49
Dict Match	2.36	2.43	1.57	1.54

Table 11: French MLM loss at 100k steps of different initializations and training protocols.

French XNLI acc.

	Embed Layers		All Layers		Best
	1e-4	5e-4	5e-5	1e-4	
Scratch	-	-	-	0.74(±0.007)	0.74(±0.007)
Embed Reinit	0.682(±0.026)	0.709(±0.003)	0.738(±0.008)	0.736(±0.004)	0.738(±0.008)
Freq Match	-	-	0.739(±0.005)	0.746(±0.004)	0.746(±0.004)
Align Match	**0.774(±0.004)**	0.759(±0.005)	0.785(±0.005)	**0.782(±0.004)**	0.785(±0.005)
Dict Match	0.766(±0.004)	**0.764(±0.007)**	**0.786(±0.001)**	0.778(±0.005)	**0.786(±0.001)**
`mBERT`	-	-	-	-	0.719 (±0.006)
`FlauBERT`	-	-	-	-	0.806

Table 12: French XNLI accuracy of initializations and training protocols. Le et al. (2019), a robustly optimized BERT, reported 0.806. `mBERT` was finetuned and tested in the French language. Bold represents the best in each column.

French PAWS-X acc.

	Embed Layers		All Layers		Best
	1e-4	5e-4	5e-5	1e-4	
Scratch	-	-	-	0.862(±0.007)	0.862(±0.007)
Embed Reinit	0.814(±0.008)	0.657(±0.123)	0.855(±0.004)	0.869(±0.013)	0.869(±0.013)
Freq Match	-	-	0.858(±0.004)	0.872(±0.003)	0.872(±0.003)
Align Match	0.806(±0.132)	0.864(±0.005)	**0.884(±0.005)**	**0.879(±0.004)**	**0.884(±0.005)**
Dict Match	**0.812(±0.135)**	**0.87(±0.002)**	0.882(±0.007)	0.874(±0.003)	0.882(±0.007)
`mBERT`	-	-	-	-	0.700(±0.006)
`FlauBERT`	-	-	-	-	0.895

Table 13: French PAWS-X accuracy of initializations and training protocols. Le et al. (2019) reported 0.895. `mBERT` was finetuned and tested in the French language. Bold represents the best in each column.

German MLM loss at 100k Steps

	Embed Layers		All Layers	
	1e-4	5e-4	5e-5	1e-4
Scratch	-	-	-	1.92
Embed Reinit	3.54	3.15	2.30	1.88
Freq Match	-	-	2.22	1.81
Align Match	2.85	2.58	1.56	1.46
Dict Match	2.53	2.87	1.79	1.56

Table 14: German MLM loss at 100k steps of different initializations and training protocols.

German XNLI acc.

	Embed Layers		All Layers		Best
	1e-4	5e-4	5e-5	1e-4	
Scratch	-	-	-	0.727(±0.003)	0.727(±0.003)
Embed Reinit	0.693(±0.02)	0.654(±0.113)	0.728(±0.005)	0.738(±0.007)	0.738(±0.007)
Freq Match	-	-	0.732(±0.006)	0.704(±0.075)	0.732(±0.006)
Align Match	**0.764(±0.023)**	**0.769(±0.007)**	**0.781(±0.006)**	**0.784(±0.005)**	**0.784(±0.005)**
Dict Match	0.729(±0.094)	0.762(±0.006)	0.775(±0.005)	0.773(±0.005)	0.775(±0.005)
mBERT	-	-	-	-	0.685(±0.029)
German BERT	-	-	-	-	0.789(±0.004)

Table 15: German XNLI accuracy of initializations and training protocols. German BERT results were 0.789 (±0.004), reproduced. mBERT was finetuned and tested in the German language. Bold represents the best in each column.

German PAWS-X acc.

	Embed Layers		All Layers		Best
	1e-4	5e-4	5e-5	1e-4	
Scratch	-	-	-	0.817(±0.007)	0.817(±0.007)
Reinit Embed	0.651(±0.095)	0.76(±0.039)	0.807(±0.007)	0.816(±0.003)	0.816(±0.003)
Freq	-	-	0.812(±0.008)	0.815(±0.012)	0.815(±0.012)
Align	0.818(±0.015)	0.822(±0.006)	**0.838(±0.008)**	**0.839(±0.006)**	**0.839(±0.006)**
Dict	**0.827(±0.008)**	**0.823(±0.007)**	0.828(±0.005)	0.809(±0.056)	0.828(±0.005)
mBERT	-	-	-	-	0.673(±0.006)
German BERT	-	-	-	-	0.828(±0.008)

Table 16: German PAWS-X accuracy of initializations and training protocols. German BERT results were 0.828 (±0.008), reproduced.mBERT was finetuned and tested in the German language. Bold represents the best in each column.