Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 1418–1427
Marseille, 20-25 June 2022

Frustratingly Easy Performance Improvements for Low-resource Setups: A Tale on BERT and Segment Embeddings

Rob van der Goot,[♣] Max Müller-Eberstein,[♣] Barbara Plank[♣◇]
[♣]Computer Science Department, IT University of Copenhagen
[◇]Center for Information and Language Processing (CIS), LMU Munich, Germany
robv@itu.dk, mamy@itu.dk, bapl@itu.dk

Abstract

As input representation for each sub-word, the original BERT architecture proposes the sum of the sub-word embedding, position embedding and a segment embedding. Sub-word and position embeddings are well-known and studied, and encode lexical information and word position, respectively. In contrast, *segment* embeddings are less known and have so far received no attention. The key idea of segment embeddings is to encode to which of the two sentences (segments) a word belongs—the intuition is to inform the model about the separation of sentences for the next sentence prediction pre-training task. However, little is known on whether the choice of segment impacts downstream prediction performance. In this work, we try to fill this gap and empirically study the impact of alternating the segment embedding during inference time for a variety of pre-trained embeddings and target tasks. We hypothesize that for single-sentence prediction tasks performance is not affected—neither in mono- nor multilingual setups—while it matters when changing the segment IDs in paired-sentence tasks. To our surprise, this is not the case. Although for classification tasks and monolingual BERT models no large differences are observed, particularly *word-level multilingual* prediction tasks are heavily impacted. For low-resource syntactic tasks, we observe impacts of segment embedding and multilingual BERT choice. We find that the default setting for the most used multilingual BERT model underperforms heavily, and a simple swap of the segment embeddings yields an average improvement of 2.5 points absolute LAS score for dependency parsing over 9 different treebanks.

Keywords: multilingual representations, BERT embeddings, segment embeddings, parsing, classification

1. Introduction

Transformer-based contextualized embeddings have led to an increase in performance on countless Natural Language Processing (NLP) tasks. However, understanding why they work, and what is encoded in their embeddings, is still an active area of research also coined "BERTology" (Rogers et al., 2020) for the most popular type of contextualized embeddings based on BERT (Devlin et al., 2019). We refer to Rogers et al. (2020) for an overview of recent work studying the inner workings of BERT and focus on a so-far ignored part of the architecture: segment embeddings.

In BERT, the input text is first tokenized using a basic punctuation/whitespace tokenizer, and then split into sub-words which stem from a vocabulary of word-pieces (Wu et al., 2016). These sub-words are then encoded as the sum of three distinct embeddings (see also Figure 1): 1) sub-word embeddings, 2) positional embeddings, 3) segment embeddings. Compared to the sub-word and positional embeddings, the intuition behind segment embeddings is less obvious, especially for unsegmented (i.e. single sentence) input (detailed in Section 2.1). While the first two embedding types differ between each vocabulary item or position, segment embeddings are identical across all sub-words of a segment and only differ if input strings containing multiple segments are fed into the model simultaneously (e.g. an input containing two sentences separated by a `[SEP]` token).

Although there has been some work into analyzing sub-words and positional embeddings (Schick and Schütze, 2020; Wang and Chen, 2020; Dufter et al., 2021; Wang et al., 2021), the effect of segment embeddings remains under-researched. In this paper, we aim to inspect the effect of the segment embeddings at inference time. Our contributions are:

- We experiment with a variety of strategies for encoding segment IDs.
- We analyze which strategies are beneficial for multiple task-types.
- We show that especially in low-resource setups the most frequently used multilingual BERT embeddings can perform substantially higher for word-level syntactic tasks (i.e. 2.5 LAS points on average for dependency parsing) by simply using the embedding for the second sentence instead of the default first.[1]
- All our code and results are publicly available at `https://bitbucket.org/robvanderg/segmentembeds/`

2. Segment Embeddings

2.1. Definition

A core component of the Transformer architecture introduced by Vaswani et al. (2017) is its ability to at-

[1]The resulting embeddings can be found at `https://huggingface.co/robvanderg/bert-base-multilingual-cased-segment1`

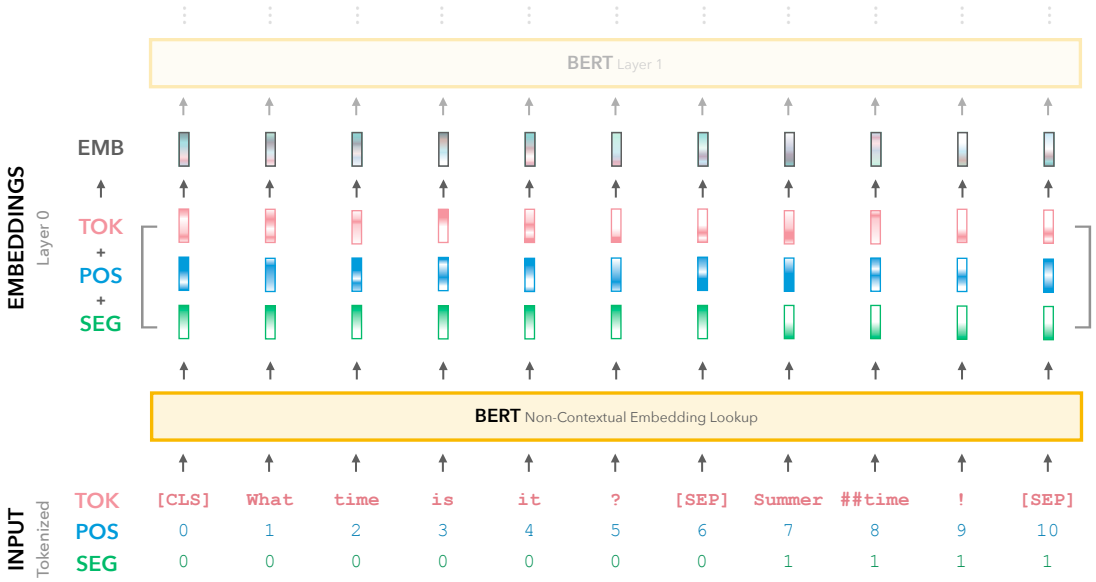

Figure 1: Schematic overview of how input is used in BERT embeddings.

tend to all inputs simultaneously. The now ubiquitous BERT architecture (Devlin et al., 2019) shares this property, computing attention maps over all input sub-words. As this removes any implicit ordering of the input, both the original Transformer and BERT include the sub-word embeddings summed with positional embeddings. Since position IDs remain the same even when sub-words at the position vary, the model learns to associate positional embeddings with the sequential nature of the input.

While absent in the original Transformer, BERT further adds segment embeddings to the non-contextualized input representation. In practice, this means that the input is combined with a sequence of segment IDs (a lookup of the sequence *"0 0 ... 0 1 ... 1 1"*), which are found in a lookup table of size 2 $\times$ embedding dimensionality (768 in $\text{BERT}_{\text{base}}$). The three embedding layers contributing to BERT's input are visualized in Figure 1. The original function of segment embeddings is not described in detail (Devlin et al., 2019), however it is clear that they are intended to help a model distinguish between two segments within the same input sequence (e.g. two sentences).

During pre-training, this behaviour is explicitly encouraged through the task of next sentence prediction (NSP): Two sentences are fed as input to BERT and the model is asked to discriminate between the true next sentence and a randomly sampled alternative sentence occurring 50% of the time. For this task, it is important for the model to be able to distinguish between the two input sentences, hence segment embeddings are used.

During subsequent task-specific fine-tuning, segment embeddings could store information relevant to the task. For example, when a relevant part of an answer should be extracted in a question answering dataset, the 0's could indicate that the segment is the question, and the 1's could indicate the segment which contains the answer. Devlin et al. (2019) indeed demonstrate that disabling NSP in BERT pre-training decreases downstream performance on tasks involving sentence pairs such as question answering and natural language inference. Later work has however challenged the usefulness of NSP (Liu et al., 2019; Yang et al., 2019; Joshi et al., 2020; Aroca-Ouellette and Rudzicz, 2020), and both NSP and segment embeddings are no longer commonly used during the training of masked language models. As BERT still remains one of the most ubiquitous architectures and since segment embeddings are also included in every available language or domain-specific BERT variant, we believe the effect of pre-trained segment embeddings to be highly relevant and worth investigating.

2.2. Segment Embeddings in Practice

In the widely used implementations of HuggingFace (Wolf et al., 2020) segment IDs are referred to as `token_type_ids` and are set to a sequence of 0s matching the length of the input by default. When two input segments (i.e. sentences) are passed to the library's BERT-tokenizer, it generates a sequence of appropriate and distinct segment IDs (i.e. 0s and 1s). However, in practice we found that some practitioners using the library set all segment IDs to 1, presumably confusing them with an attention mask.

Furthermore, it should be noted that the segment embeddings are implemented at the base class level, even though they are not available for all language models. This means that even for language models that do not have pre-trained segment embeddings, by default they are automatically trained during fine-tuning (note that there could also be only one segment ID). This highlights that segment embeddings are even more widespread than one might assume.

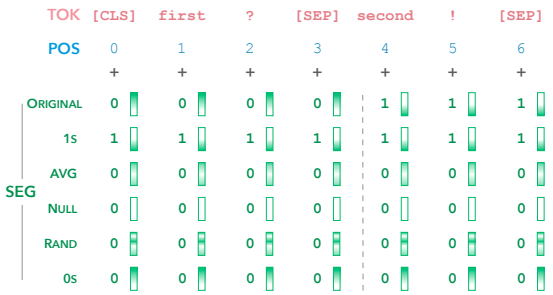

Figure 2: Visualization of the segment alternations.

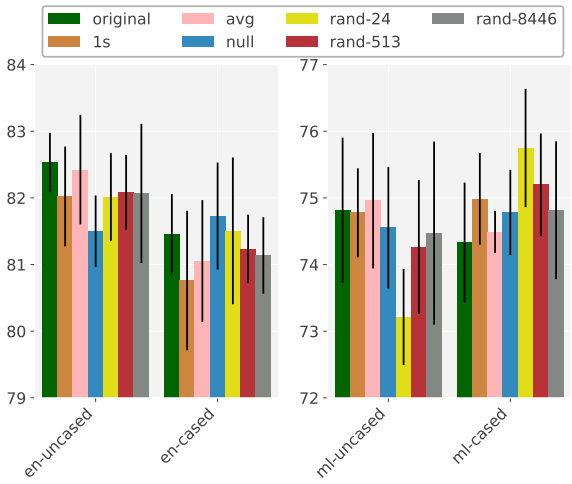

Figure 3: Average accuracies for single-sentence classification on the dev data in a low-resource setup.

3. Experiments

3.1. Models

In order to explore the impact of segment information on downstream performance, we study several segment embedding-specific alternations to the original setup (visualized in Figure 2):

ORIGINAL: For tasks consisting of two sentences, we experiment with a setup similar to the one used during training, using 0 for sub-words in the first sentence, and 1 for sub-words in the second sentence. For single sentence tasks this means only 0 is used.

1S: Uses 1 as segment ID for all sub-words. This sub-word + position + segment combination is unseen during pre-training as the second segment never occurs at the beginning of the input.

AVG: Uses the mean of both segment embeddings for all sub-words, also for tasks where the input consists of a single sentence.

NULL: Sets all values in the segment embeddings to 0 for all inputs. This can be seen as an ablation where segment embeddings are disabled.

RAND*: Each segment embedding is re-initialized randomly using three seeds. Although the Wolf et al. (2020) implementation samples a standard normal distribution, we re-implement the original truncated normal distribution to ensure complete parity.

0S: Uses 0 as segment ID for all sub-words. This setting is unseen during pre-training on sentence-pair tasks. In single-sentence tasks this setting is equivalent to ORIGINAL and therefore not reported separately.

For each segment embedding alternation, we explore the differences between monolingual BERT and multilingual BERT (mBERT) as well as their cased ($\text{BERT}_{\text{case}}$, $\text{mBERT}_{\text{case}}$) and uncased variants ($\text{BERT}_{\text{unc}}$, $\text{mBERT}_{\text{unc}}$). For the monolingual models we use the $\text{BERT}_{\text{base}}$ variant (compared to $\text{BERT}_{\text{large}}$) as its architecture also forms the basis for the multilingual model. Although for English both the cased and uncased versions are widely used, it should be noted that use of the multilingual $\text{mBERT}_{\text{unc}}$ is actually discouraged by its maintainers.
Specifically relevant to segment embeddings, we further identified that the NSP task during mBERT pre-training may involve two sentences from different languages (we deduced this information from the code, see Appendix A for details). We hypothesize that this could affect the learning process of the segment embeddings as the NSP task becomes substantially easier. Differences between the monolingual and multilingual models therefore warrant extra attention.

3.2. Setup

We run our experiments using MaChAmp v0.2 (van der Goot et al., 2021) with default hyperparameters. The experiments are divided into three different types of tasks: sentence-level and sentence-pair tasks (Section 4.1) as well as word-level tasks (Section 4.2), wherein the specific tasks are the following:
For our sentence-level tasks, we use SST-2 (Socher et al., 2013) and CoLa (Warstadt et al., 2019) from the GLUE benchmark (Wang et al., 2018), which are sentiment and linguistic acceptability detection datasets respectively.
As sentence-pair tasks, we use all GLUE tasks which consist of two sentences per input (Williams et al., 2018; Dolan and Brockett, 2005; Rajpurkar et al., 2018; Dagan et al., 2006; Bar Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009; Bowman et al., 2015; Levesque et al., 2011).[2] All classification tasks are evaluated with accuracy.
For our word-level tasks, we use the Universal Dependencies data (Nivre et al., 2020), more specifically, we use the set of treebanks sampled by Smith et al. (2018), which was "chosen to reflect a diversity of writing systems, character set sizes, and morphological complexity". For the English embeddings, we only use the EWT treebank (Silveira et al., 2014). We include UPOS tagging, morphological tagging, lemmatization and dependency parsing, all with the default

[2] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

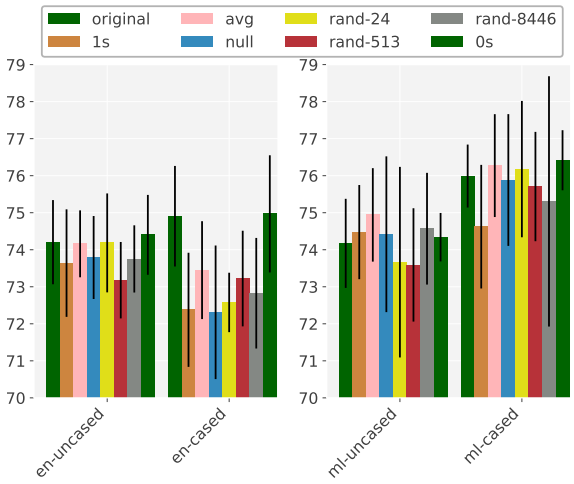

Figure 4: Average accuracies for classification on sentence pairs on the dev data in a low-resource setup.

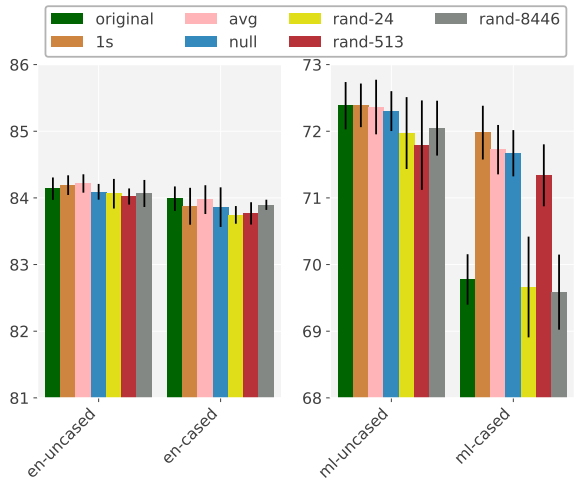

Figure 5: Average LAS scores for each setting (Section 3) on the dev data in a low-resource setup. The mono-lingual embeddings are results only for EWT, the multilingual embeddings are averages over 9 treebanks. Black lines indicate standard deviation.

task-types/hyperparameters of MaChAmp in a multi-task setup where each task has its own decoder. We assume gold tokenization for all experiments, and remove multi-word tokens with UD-conversion-tools.[3] In Section 4.2, we focus on dependency parsing and report labeled attachment scores (LAS) following Zeman et al. (2018), while for the remaining tasks, we report accuracy in Appendix D.

We follow van der Goot (2021), and use a tune-set for early stopping to avoid overfitting on the development or test data. In our early experiments we saw that performance differences are largest in early epochs (see also Figure 7), which is why we hypothesize that segment embeddings have the largest effect in low-resource setups. To save compute, we therefore perform our main experiments on a low-resource setup (10% of training data for UD, and 1,000 training instances for the classification tasks), before applying the most interesting setups on the full data (Section 4.3). Reported results are averages over five runs with different random seeds. We report only results on the development data, as this is an analysis paper and we would like to avoid overusing the test data. It should be noted that varying minimum/maximum limits are used on the y-axes of our figures, but that the size of the ranges are consistent within figures.

4. Results

4.1. Classification Results

For the single-sentence classification tasks (Figure 3), the differences between the models are mostly within one standard deviation. We see slightly higher scores across segment alternations of English BERT_{unc}. The largest difference is observed for one of the RAND* models in the multilingual setup, where one of the seeds (24) leads to the lowest performance for $\text{mBERT}_{\text{unc}}$ and the best performance for $\text{mBERT}_{\text{case}}$ relative to the ORIGINAL setup. This shows that fluctuations in the random initialization of segment embeddings can have large impacts on downstream results.

For the sentence-pair tasks (Figure 4), the standard deviations are larger, making the overall effect of segment embeddings less clear/harder to measure. Most performance differences are within one standard deviation, leading us to conclude that it seems to not be very important how the segment embeddings are initialized in this setup.

4.2. Dependency Parsing Results

The most striking results are shown in Figure 5 for the multilingual dependency parsing tasks, where the largest differences are observed for the highly popular $\text{mBERT}_{\text{case}}$. This is the most frequently downloaded multilingual language model on the HuggingFace Models platform, and strikingly, the default setup results in one of the worst performances. Compared to the ORIGINAL setup, a simple flip of segment IDs from 0 to 1 for all input sub-words (1S) increases performance by up to 2.5 LAS — a non-standard combination unseen during the extensive pre-training process. Similarly, averaged (AVG) or empty (NULL) segment embeddings as well as the re-initialized RAND-513 also substantially outperform the ORIGINAL setup.

Another perhaps surprising find is that $\text{mBERT}_{\text{unc}}$ outperforms $\text{mBERT}_{\text{case}}$, although the uncased model is "not recommended" and has inferior normalization according to the official GitHub release page.[4] A closer inspection shows that this is mostly due to the Ancient

[3] https://github.com/bplank/ud-conversion-tools

[4] https://github.com/google-research/bert/blob/master/multilingual.md

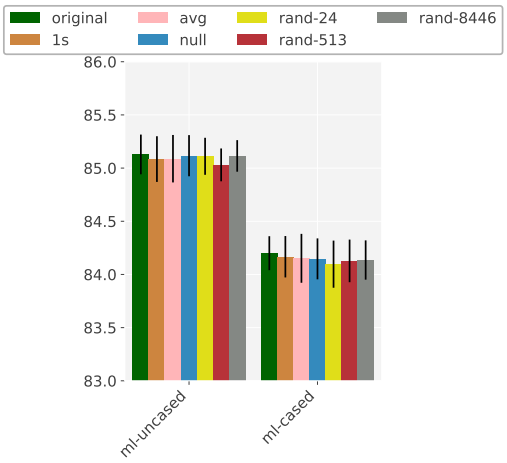

Figure 6: Average LAS scores for each setting (Section 3) on the dev data when training on full training splits. The mono-lingual embeddings are results only for EWT, the multilingual embedding results are averages over 9 treebanks.

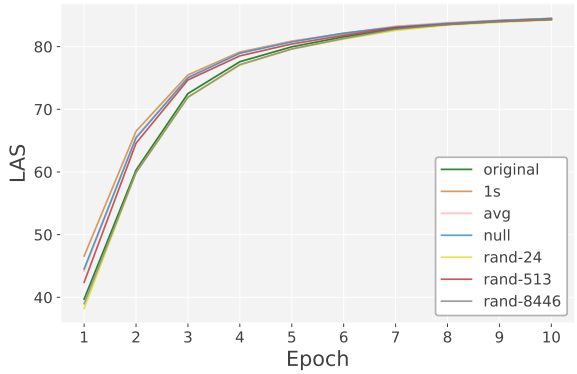

Figure 7: Average LAS scores for $\text{mBERT}_{\text{case}}$ on all treebanks for the first 10 epochs (of 20) during training on the tune data when training on the full treebanks.

Greek PROIEL treebank (see Appendix C). Unfortunately, the exact details of the pre-training differences between cased and uncased mBERT are not reported, so the main reason for these performance differences remain unknown.

In terms of segment information learned during pre-training, the fact that the provided embeddings (ORIGINAL, 1S and AVG) perform better than the RAND* or NULL strategies for all models except for $\text{mBERT}_{\text{case}}$ indicates that useful information is stored in the segment embeddings.

In Appendix D we provide additional results for other UD tasks: UPOS tagging, XPOS tagging, morphological tagging and normalization. Although the ranges of the scores differ, the trends closely resemble the ones we see for dependency parsing. We conclude that for these syntactic tasks in low-resource multilingual setups, segment embeddings in the most commonly used $\text{mBERT}_{\text{case}}$ have a noticeable effect on downstream performance — e.g., increasing scores improve substantially by simply flipping segment ID 0 to 1.

4.3. High-resource Versus Low-resource

Considering the large effect of segment embeddings on $\text{mBERT}_{\text{case}}$ in low-resource settings and given its widespread use, we re-ran the experiments using the full training data. Results on dependency parsing (Figure 6) show that the performance differences disappear, as all models perform within one standard deviation of each other. This confirms our prior hypothesis that the effect of the few segment embedding parameters are most pronounced in low-resource settings.

When inspecting the per-epoch performance during fine-tuning (Figure 7), we see that the differences are still apparent in the first epochs and vanish later. This indicates that when fine-tuning for long enough, the parameters stored in the segment embeddings are optimized and their initialization is less relevant. This further supports the finding that the effect of using pre-initialized segment embeddings is only important for low-resource setups.

5. Discussion

We investigated the question on how much segment embeddings in widely-used, pre-trained BERT models affect downstream performance of NLP tasks. Despite using combinations of sub-word, position and segment embeddings which are non-standard and unseen during pre-training, we found the performance differences for classification tasks and monolingual BERT models to be relatively modest. This invariance to segment embedding alterations is consistent for sentence-pair tasks which were previously shown to benefit from segment-specific information (Devlin et al., 2019).

In contrast, we found the most frequently used multilingual model, $\text{mBERT}_{\text{case}}$, to show a striking performance difference in low-resource setups for word-level tasks. In this setup, the ORIGINAL setup is outperformed by a simple swap of segment IDs yielding an average 2.5 absolute LAS performance increase across a variety of multilingual treebanks. Strangely enough this difference only occurs for the cased model, and unfortunately the exact differences between this model and the deprecated uncased model remain unclear.

One other takeaway is that even though there has been much work on how/why BERT-like models perform well, there are still parts that are heavily under-explored and not well understood. As we have seen for low-resource parsing, seemingly small architectural choices can result in substantial performance differences.

6. Acknowledgements

We would like to thank John Hewitt for the insightful discussion about segment embeddings. This research is supported by the Independent Research Fund Denmark (Danmarks Frie Forskningsfond; DFF) grant number 9063-00077B.

7. Bibliographical References

Aroca-Ouellette, S. and Rudzicz, F. (2020). On losses for modern language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4970–4981, Online, November. Association for Computational Linguistics.

Bar Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. (2006). The second PASCAL recognising textual entailment challenge.

Bentivogli, L., Dagan, I., Dang, H. T., Giampiccolo, D., and Magnini, B. (2009). The fifth PASCAL recognizing textual entailment challenge.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.

Dagan, I., Glickman, O., and Magnini, B. (2006). The PASCAL recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pages 177–190. Springer.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (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, June. Association for Computational Linguistics.

Dolan, W. B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Dufter, P., Schmitt, M., and Schütze, H. (2021). Position information in transformers: An overview. *CoRR*, abs/2102.11090.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. (2007). The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9. Association for Computational Linguistics.

Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. (2020). SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Levesque, H. J., Davis, E., and Morgenstern, L. (2011). The Winograd schema challenge. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, volume 46, page 47.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Nivre, J., de Marneffe, M.-C., Ginter, F., Hajič, J., Manning, C. D., Pyysalo, S., Schuster, S., Tyers, F., and Zeman, D. (2020). Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France, May. European Language Resources Association.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July. Association for Computational Linguistics.

Rogers, A., Kovaleva, O., and Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Schick, T. and Schütze, H. (2020). BERTRAM: Improved word embeddings have big impact on contextualized model performance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3996–4007, Online, July. Association for Computational Linguistics.

Silveira, N., Dozat, T., de Marneffe, M.-C., Bowman, S., Connor, M., Bauer, J., and Manning, C. (2014). A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2897–2904, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Smith, A., de Lhoneux, M., Stymne, S., and Nivre, J. (2018). An investigation of the interactions between pre-trained word embeddings, character models and POS tags in dependency parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2711–2720, Brussels, Belgium, October-November. Association for Computational Linguistics.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October. Association for Computational Linguistics.

van der Goot, R., Üstün, A., Ramponi, A., Sharaf, I., and Plank, B. (2021). Massive choice, ample tasks (MaChAmp): A toolkit for multi-task learning in NLP. In *Proceedings of the 16th Conference of the European Chapter of the Association for Compu-*

tational Linguistics: System Demonstrations, pages 176–197, Online, April. Association for Computational Linguistics.

van der Goot, R. (2021). We need to talk about train-dev-test splits. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4485–4494, Online and Punta Cana, Dominican Republic, November. Association for Computational Linguistics.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Wang, Y.-A. and Chen, Y.-N. (2020). What do position embeddings learn? an empirical study of pre-trained language model positional encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6840–6849, Online, November. Association for Computational Linguistics.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November. Association for Computational Linguistics.

Wang, B., Shang, L., Lioma, C., Jiang, X., Yang, H., Liu, Q., and Simonsen, J. G. (2021). On position embeddings in {bert}. In *International Conference on Learning Representations*.

Warstadt, A., Singh, A., and Bowman, S. R. (2019). Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, March.

Williams, A., Nangia, N., and Bowman, S. (2018). 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)*, pages 1112–1122, New Orleans, Louisiana, June. Association for Computational Linguistics.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October. Association for Computational Linguistics.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., Nivre, J., and Petrov, S. (2018). CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 1–21, Brussels, Belgium, October. Association for Computational Linguistics.

A. Is mBERT Pre-trained on Cross-lingual Sentence Pairs?

We did not succeed in finding/obtaining information from the original authors on whether mBERT's next sentence prediction was done on multiple languages within one sentence pair. However, inspection of the code led us to believe that the sentences are in many cases from two different languages. Below we explain our argumentation.

The BERT code[5] allows for a list of input files. These could be either 1) a separate file for each language 2) one file with a concatenation of all languages. In 2) a random sentence is picked from the concatenation, which in many cases will be from another language. For 1), all separate files are concatenated and shuffled[6] before the actual random sentences are generated in the code.[7]

B. Implementation Details

The pre-trained language models used in our experiments are provided in HuggingFace Models (Wolf et al., 2020) under the identifiers:

- `bert-base-cased`
- `bert-base-uncased`
- `bert-base-multilingual-cased`
- `bert-base-multilingual-uncased`

We further use MaChAmp v0.2 (van der Goot et al., 2021) with all default settings.

C. Scores per Treebank

In Figure 8- 16 we report the scores obtained on the single treebanks, again averaged over 5 seeds. Although not used in the paper, we include results of the English $\text{BERT}_{\text{base}}$ on the other non-English target languages in these graphs.

D. Scores for Other UD Tasks

We also report accuracy scores for the other tasks MaChAmp performs: UPOS (Figure 17), XPOS (Figure 18), morphological tagging (Figure 19) and lemmatization (Figure 20).

[5] https://github.com/google-research/bert/blob/master/create_pretraining_data.py

[6] https://github.com/google-research/bert/blob/master/create_pretraining_data.py#L183

[7] https://github.com/google-research/bert/blob/master/create_pretraining_data.py#L280

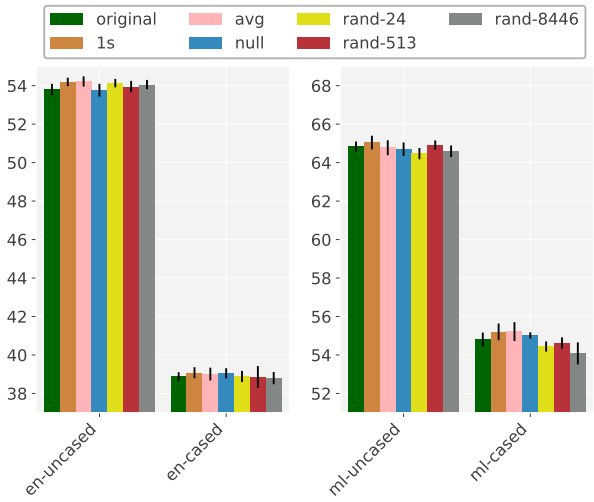

Figure 8: UD_Ancient_Greek-PROIEL

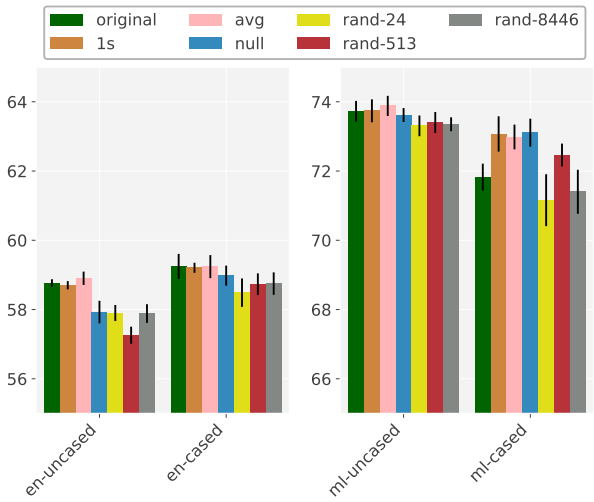

Figure 9: UD_Arabic-PADT

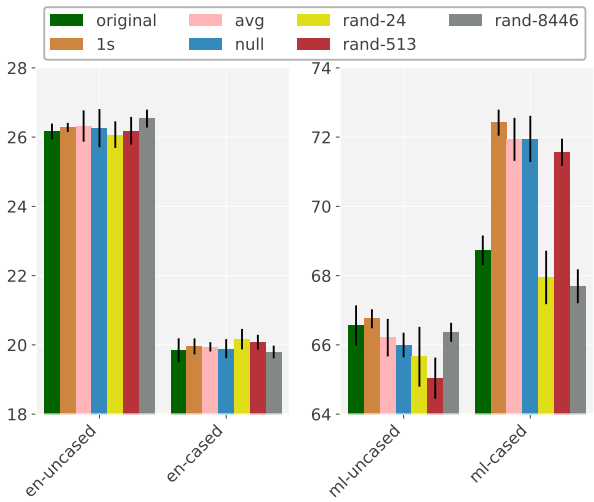

Figure 10: UD_Chinese-GSD

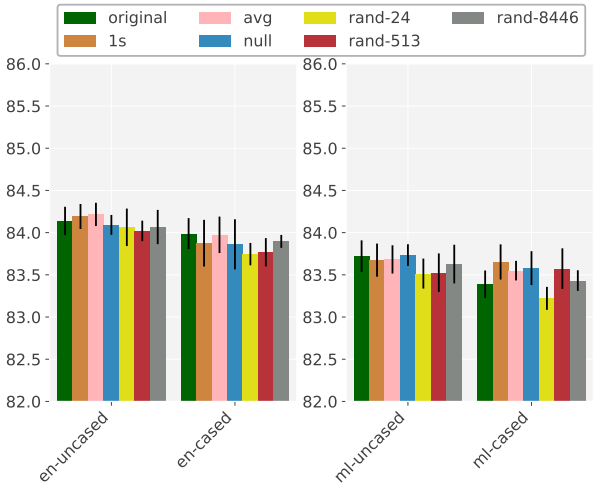

Figure 11: UD_English-EWT

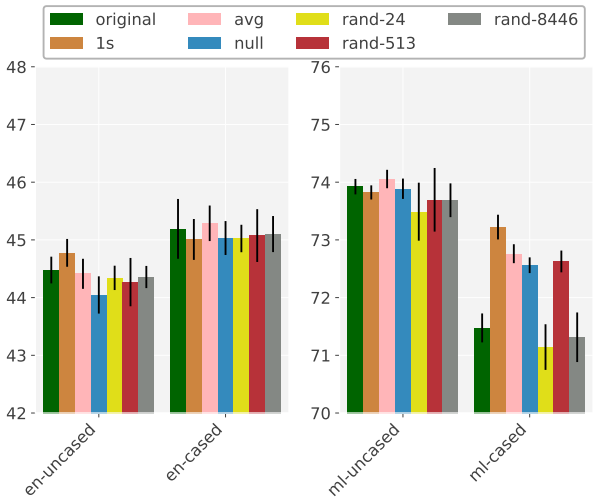

Figure 12: UD_Finnish-TDT

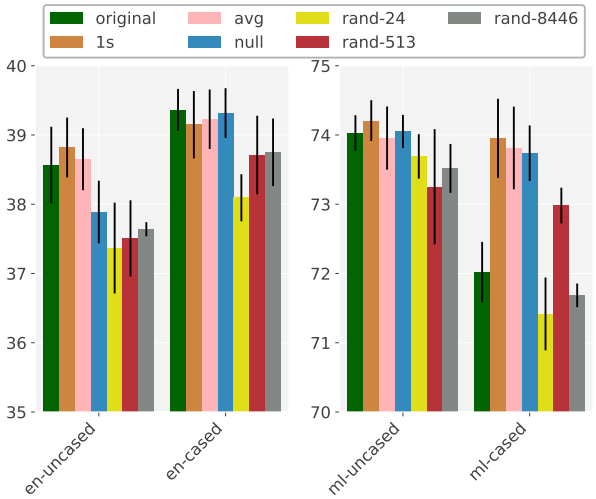

Figure 13: UD_Hebrew-HTB

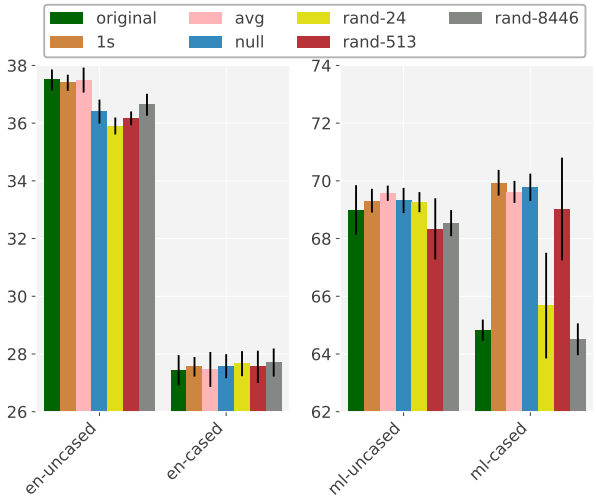

Figure 14: UD_Korean-GSD

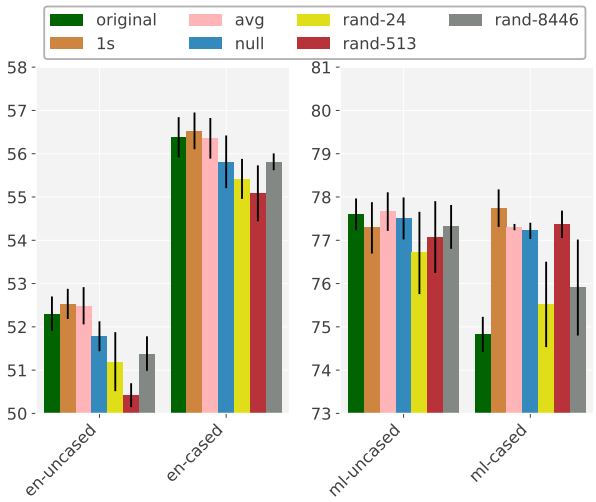

Figure 15: UD_Russian-GSD

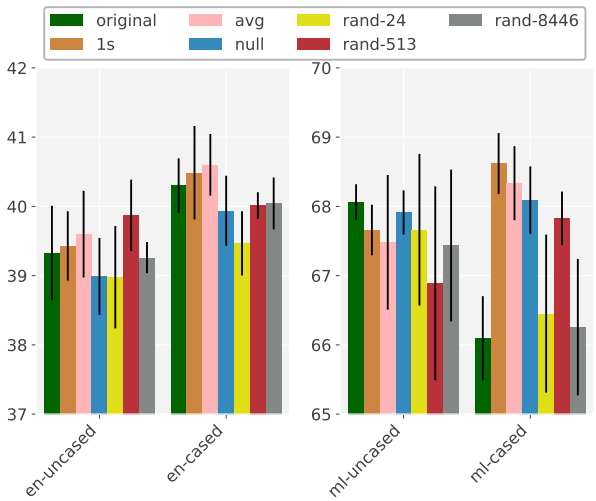

Figure 16: UD_Swedish-Talbanken

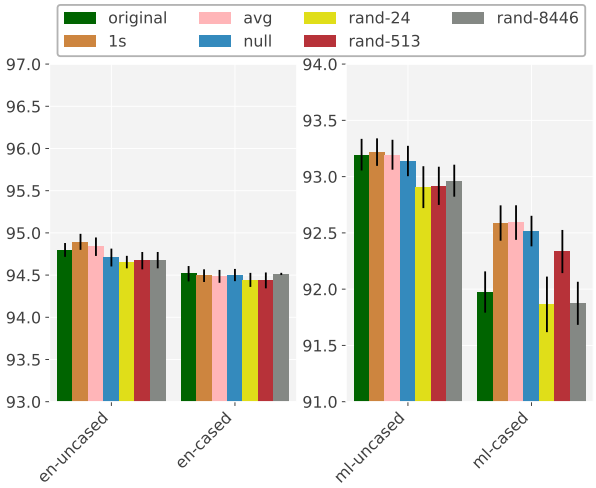

Figure 17: UPOS

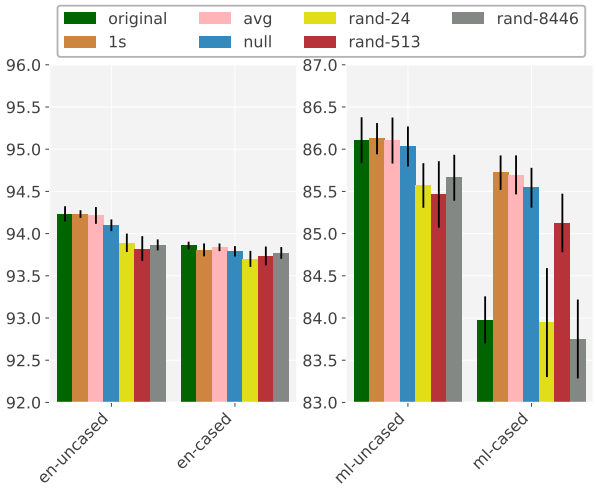

Figure 18: XPOS

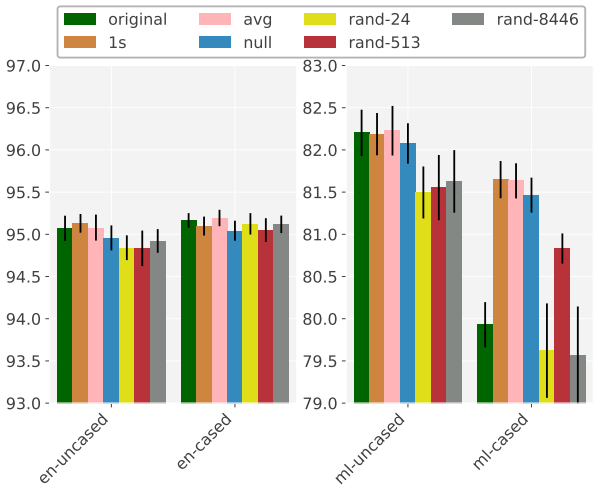

Figure 19: Morphological features

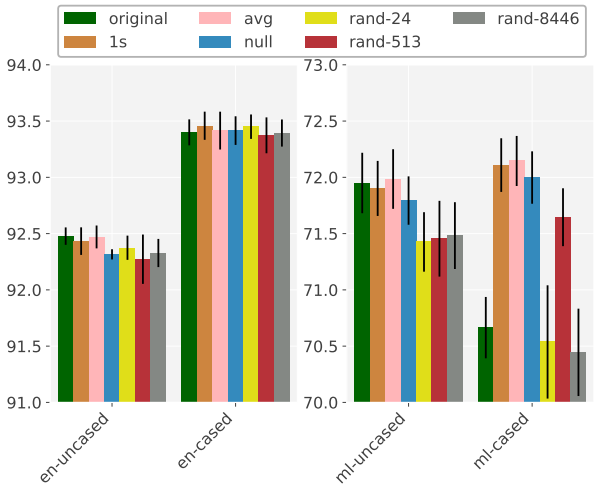

Figure 20: Lemmatization