Parsing with Pretrained Language Models, Multiple Datasets, and Dataset Embeddings

Rob van der Goot
IT University of Copenhagen
robv@itu.dk

Miryam de Lhoneux
Uppsala University
KU Leuven
University of Copenhagen
ml@di.ku.dk

Abstract

With an increase of dataset availability, the potential for learning from a variety of data sources has increased. One particular method to improve learning from multiple data sources is to embed the data source during training. This allows the model to learn generalizable features as well as distinguishing features between datasets. However, these dataset embeddings have mostly been used before contextualized transformer-based embeddings were introduced in the field of Natural Language Processing. In this work, we compare two methods to embed datasets in a transformer-based multilingual dependency parser, and perform an extensive evaluation. We show that: 1) embedding the dataset is still beneficial with these models 2) performance increases are highest when embedding the dataset at the encoder level 3) unsurprisingly, we confirm that performance increases are highest for small datasets and datasets with a low baseline score. 4) we show that training on the combination of all datasets performs similarly to designing smaller clusters based on language-relatedness.[1]

1 Introduction

Many studies have shown the benefits of training dependency parsers jointly for multiple treebanks, either within the same language (Stymne et al., 2018) or across different languages (multilingual models; Ammar et al., 2016; Vilares et al., 2016; Smith et al., 2018), which makes it possible to transfer knowledge between treebanks. This has been enabled by the development of treebanks in multiple languages annotated according to the same guidelines which have been released by the Universal Dependencies (UD; Nivre et al., 2020) project. In this context, a method which has been shown to be effective is the use of embeddings that represent each individual treebank. This is referred to as a *language embedding* (Ammar et al., 2016) or a *treebank embedding* (Smith et al., 2018), we opt for the more general term: *dataset embedding*. The main intuition behind these dataset embeddings is that they allow the model to learn useful commonalities between different datasets, while still encoding dataset specific knowledge.

In the last few years, large multilingual pretrained language models (LMs) pretrained on the task of language modelling, such as mBERT (Devlin et al., 2019) or XLM-R (Conneau et al., 2020), have made it possible to train high performing multilingual models for many different NLP tasks, including dependency parsing (Kondratyuk and Straka, 2019). These models have shown surprising cross-lingual abilities in spite of not getting any cross-lingual supervision. Wu and Dredze (2019), for example, fine-tuned mBERT on English data for five different tasks and applied it to different languages with high accuracy, relative to the state-of-the-art for these tasks. These large pretrained multilingual LMs are now widely used in multilingual NLP.

Dataset embeddings have so far only been evaluated in the context of multilingual dependency parsing without such large pretrained multilingual LMs. Given the large gains in accuracy that have been obtained by using them, and given that they seem to be learning to do cross-lingual transfer without cross-lingual supervision, it is unclear whether or not datataset embeddings can still be useful in this

[1]Code available at: https://bitbucket.org/robvanderg/dataembs2, our implementations will also be included in the next MaChAMp release (v0.3)

context. This is the question we ask in this paper. Our main research question can be formulated as follows:

RQ Is the information learned by dataset embeddings complementary to the information learned by large multilingual LM-based parsers?

2 Background

Vilares et al. (2016) were among the first to exploit UD treebanks to train models for multiple languages. They trained parsing models on 100 pairs of languages by simply concatenating the treebank pairs. They observed that most models obtained comparable accuracy to the monolingual baseline and some outperformed it. This shows that even in its simplest form, multilingual training can be beneficial. Ammar et al. (2016) build a more complex model to train a parser for eight languages. They introduce a vector representation of each language which is used as a feature, concatenated to the word representations of each word in the sentence.

Smith et al. (2018) used this idea in the context of the CoNLL 2018 shared task (Zeman et al., 2018) where the task was to parse test sets in 82 languages. Smith et al. (2018) found that training models on clusters of related languages using dataset embeddings led to a substantial accuracy gain over training individual models for each language.

Stymne et al. (2018) further exploited this dataset embedding method in the monolingual context, using heterogeneous treebanks. They compared this method to several methods including simply concatenating the treebanks to learn a unique parser for all treebanks and found the dataset embedding method to be superior to all other methods.

Wagner et al. (2020) investigated whether dataset embeddings can be useful in an out-of-domain scenario where the treebank of the target data is unknown. They found that it is possible to predict dataset embeddings for such target treebanks, making the method useful in this scenario.

Kondratyuk and Straka (2019) trained a single model for the 75 languages available in UD at the time by concatenating all treebanks and using a large pretrained LM. They found that this did not hurt accuracy compared to training monolingual models, and it even improved accuracy for some languages. This type of model has become standard and has been used in many studies. They did not make use of dataset embeddings in this setup.

Dataset embeddings have mostly been used with BiLSTM parsers. This may partially be due to the fact that they were concatenated to the word embedding before passing it into the encoder, which is non-trivial in LM-based setups where the word embedding and encoding size is fixed. To the best of our knowledge, the only attempt to use dataset embeddings in combination with a LM-based parser was from van der Goot et al. (2021). They use a large pretrained multilingual LM as encoder, and concatenate the dataset embeddings to the word embedding before decoding. They show that this leads to improved performance if the task is the same and the languages/domains of the datasets differ. For a setup where different tasks are combined via multi-task learning (i.e. GLUE), dataset embeddings helped to decrease the performance gap compared to single-task models.

Contributions In this work, 1) we introduce a method to incorporate dataset embeddings also in the encoder in LM-based parsers; 2) we test whether or not dataset embeddings are useful when used in combination with large pretrained multilingual LMs, using both our newly proposed method as well as the existing approach; 3) we compare the effectiveness of dataset embedding when training on small clusters of datasets as well as training on all considered datasets simultaneously.

3 Methodology

3.1 Methods

In most previous implementations of dataset embeddings, the dataset embedding is concatenated to the word embedding before it is passed into the encoder. When using the language model as encoder, as is now commonly done with BERT-like embeddings, this is impossible, as the word embedding is expected

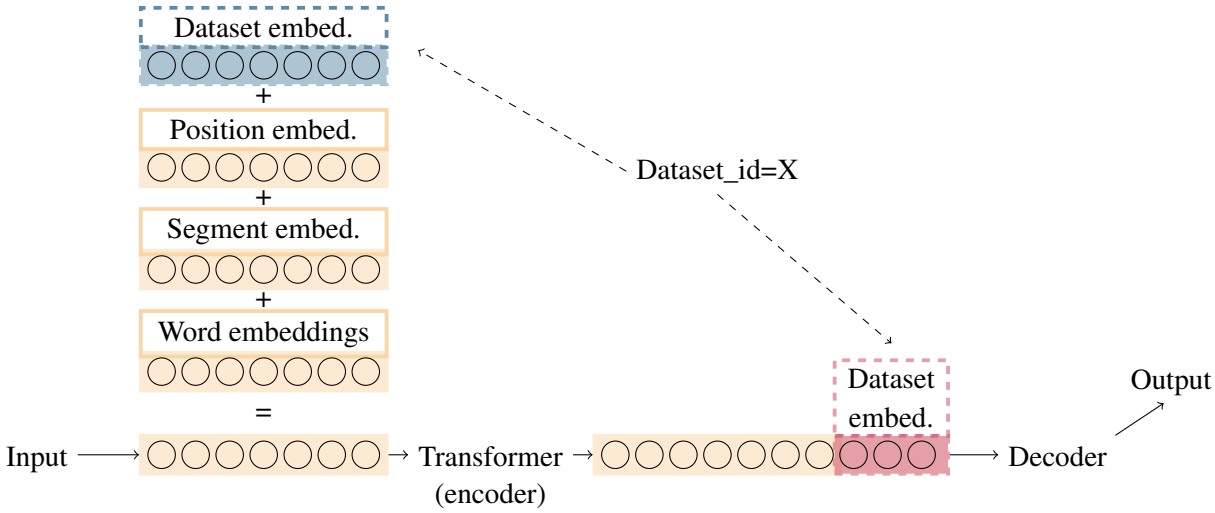

Figure 1: Visualization of the two models to integrate dataset embeddings into transformer based parsers. The dataset embeddings for DECODER is displayed with the red box (dashed on the right), and ENCODER is displayed in the blue box (dashed on the left).

to be of a fixed size. For this reason, we experiment with two alternative setups: concatenating the embedding after the encoding, and summing it to the word embedding. These two approaches are illustrated in Figure 1, and are explained in detail in the following two paragraphs. Note that these approaches can also be used simultaneously, which constitutes our third setup (BOTH).

We use the deep biaffine parser (Dozat and Manning, 2017) implementation of MaChAmp (van der Goot et al., 2021) as a framework for evaluating our models. MaChAmp already includes an implementation of dataset embeddings on the decoder level. In this implementation the output of the LM for each wordpiece is concatenated to the dataset embedding before it is passed on to the decoder. We refer to this approach as DECODER. In this setup, we choose to use dataset embeddings of size 12, based on previous work (Ammar et al., 2016; Smith et al., 2018). This embedding is then concatenated to the output of the transformer, meaning that the input size of the decoder will be the size of the original output + 12.

The second approach incorporates the dataset embeddings in the LM parameters. In most transformer-based LM implementations, multiple embeddings are summed before the transformer layers to represent the input for each wordpiece. In the BERT model for example, these are the token embeddings, segment embeddings and position embeddings. We supplement these by also summing a dataset embedding to this input. In this way, the dataset embeddings can be taken into account throughout the transformer layers. We refer to this approach as ENCODER. In this setup, we choose to match the dimension size of the embeddings at the token level, to avoid having smaller embeddings summing only to an arbitrary subset of the weights.

3.2 Experimental Setup

We essentially reproduce the experiments from Smith et al. (2018) using large pretrained multilingual LMs but do a more large-scale evaluation of the method, by testing more settings, comparing to more baselines and doing more extensive analysis. More specifically, we use the clusters from Smith et al. (2018), but use the updated versions of the treebanks (UD v2.8), and implement all models using MaChAmp, the library by van der Goot et al. (2021). We use all default hyperparameters, including the mBERT embeddings (Devlin et al., 2019) which is used during the original tuning of MaChAmp van der Goot et al. (2021). All reported results are the average over 5 runs with different random seeds for MaChAmp.

To avoid overfitting on the development or test set, we follow van der Goot (2021), and compare our models on the development split while using a tune split for model picking. We will confirm our main findings on the test data in Section 4.3. We use the updated splitting strategy proposed by van der Goot

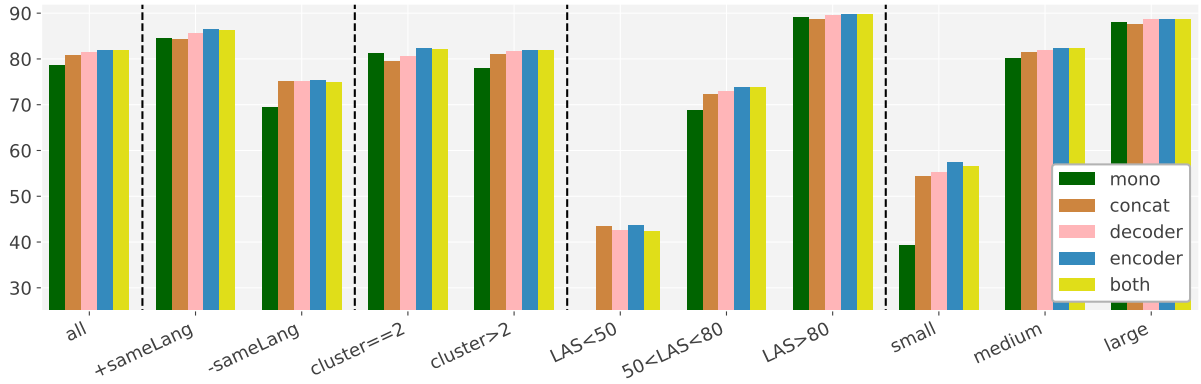

Figure 2: Average LAS scores (dev) over different subsets of the data. `All`: all data, `+sameLang`: datasets for which another in-language treebank exists, `cluster==2`: clusters of size 2, `LAS<50`: treebanks for which the 'mono´ baseline scores <50 LAS, `small, medium, large`: datasets with a maximum size of respectively: 1,000, 10,000, 20,000 sentences.

(2021): for datasets with less then 3,000 sentences, we use 50% for training, 25% for tune, and 25% for dev, for larger datasets we use 750 sentences for dev and tune, and the rest for train. We limit the size of each dataset to 20,000.

We compare models with dataset embeddings to baselines where we use the exact same parser (Figure 1), but without enabling any dataset emebddings. We use two training setups for our baselines: 1) monolingual models (MONO) and 2) models where all treebanks are concatenated (CONCAT). This is in contrast to Smith et al. (2018) who only compared to a monolingual baseline. We test this on the 59 test sets that are part of a cluster.[2] Furthermore, we also explore what happens if we train one parser on all the datasets simultaneously, similar to Udify (Kondratyuk and Straka, 2019), who do not use dataset embeddings in their setup.

4 Evaluation

4.1 Results

Full results can be found in Table 1. We can see that dataset embeddings still seem to be largely useful, outperforming the monolingual baseline (MONO) in almost all cases (56/59 test sets) and the concatenated baselines (CONCAT) in a majority of cases (40/59). The dataset embedding methods are on average 3 LAS points above the MONO baseline. These gains are even larger than the gains observed in Smith et al. (2018) which indicates that dataset embeddings are still relevant when using large pretrained multilingual LMs. It should be noted though, that a large portion of this gain is already present when using the CONCAT strategy. ENCODER scores highest overall, but for some clusters, DECODER is competetive (af-de-nl, es-ca, it, old, sw-sla). These clusters have in common that they are small and/or contain relatively high-resource languages (which are probably better represented in the mBERT embeddings).

For treebanks from languages not used during mBERT pretraining, scores are very low for the MONO baseline, but they gain a lot from training on other languages. Kazakh also has very low scores, this is probably because the training split is very small. It gains a lot already in CONCAT, likely because Turkish is a related language, but then loses accuracy when using dataset embeddings compared to CONCAT, likely because there is not enough in-language data to learn an accurate dataset embedding.

4.2 Results on Subsets

To find trends in our results, we report scores over different subsets of the data. In Figure 2, we report the results when training a parser for each cluster. The leftmost part of the graph shows the scores averaged over all datasets. `+sameLang` are the average scores for all datasets for which a dataset in the

[2]Smith et al. (2018) trained mono-treebank models for the remaining test sets

			Trained on clusters				Trained on all			
Cluster	Treebank	MONO	CONCAT	DECODER	ENCODER	BOTH	CONCAT	DECODER	ENCODER	BOTH
af-de-nl	af_afribooms	80.63	82.12	82.62	81.17	80.93	82.98	**83.53**	82.62	82.37
	nl_alpino	92.75	93.01	92.97	92.50	**93.15**	92.65	92.59	93.08	92.76
	nl_lassysmall	87.25	89.71	89.91	89.59	89.69	89.45	**90.29**	90.11	90.05
	de_gsd	87.02	87.60	87.52	87.09	87.31	87.63	**87.63**	87.25	87.53
e-sla	ru_syntagrus	**94.75**	94.74	94.56	94.60	94.67	94.50	94.45	94.51	94.52
	ru_taiga	74.93	74.54	75.11	75.75	76.21	75.43	75.95	**76.46**	76.45
	uk_iu	88.56	89.96	89.97	89.73	89.56	90.28	**90.67**	90.58	90.26
en	en_ewt	89.34	88.90	89.06	**89.67**	89.12	88.49	88.16	88.86	88.94
	en_gum	90.38	89.06	90.17	90.74	**90.91**	88.64	89.65	90.53	90.71
	en_lines	86.57	84.72	84.77	87.00	87.38	84.70	85.08	87.41	**87.42**
es-ca	ca_ancora	93.33	93.66	93.56	93.42	**93.76**	93.45	93.43	93.46	93.44
	es_ancora	92.99	93.08	**93.38**	93.30	93.30	93.35	93.28	93.24	93.16
finno	et_edt	**85.61**	84.66	85.29	85.11	85.33	85.17	84.98	85.58	85.58
	fi_ftb	89.03	82.22	89.53	**90.14**	89.87	80.15	89.01	89.06	89.38
	fi_tdt	88.55	82.59	88.99	88.90	89.35	84.24	89.13	**89.44**	89.39
	sme_giella*	49.86	62.35	62.74	62.96	64.71	59.55	59.55	63.95	**65.28**
fr	fr_gsd	94.45	**94.62**	94.40	94.57	94.46	94.49	94.42	94.44	94.54
	fr_sequoia	88.55	85.85	87.86	91.32	90.91	86.17	86.46	91.68	**91.73**
	fr_spoken	79.15	83.80	83.33	84.30	83.89	83.47	83.20	84.09	**84.50**
indic	hi_hdtb	92.54	92.47	92.56	**92.66**	92.37	92.49	92.37	92.51	92.34
	ur_udtb	81.02	81.78	81.93	**82.01**	81.74	81.66	81.41	81.77	81.42
iranian	kmr_mg*	21.43	14.29	16.67	**33.33**	30.95	21.43	19.05	28.57	14.29
	fa_seraji	**87.01**	86.96	86.84	86.37	86.28	86.32	86.30	86.54	86.27
it	it_isdt	92.92	93.33	93.22	92.96	93.09	**93.44**	93.26	93.24	93.29
	it_postwita	79.66	80.00	80.34	80.34	80.26	79.96	80.08	**80.72**	80.16
ko	ko_gsd	**82.80**	71.46	79.63	82.71	82.24	69.41	79.13	82.55	82.77
	ko_kaist	87.05	81.52	86.52	**87.66**	87.54	81.12	85.81	87.07	86.86
n-ger	da_ddt	86.24	86.37	86.32	**87.05**	86.50	85.90	85.30	86.41	86.31
	no_bokmaal	93.91	94.18	**94.77**	94.31	94.26	94.17	94.28	94.15	94.30
	no_nynorsk	92.57	92.85	93.21	**93.30**	92.93	92.73	92.47	92.93	93.15
	no_nynorsklia	74.23	76.72	77.18	76.94	77.26	76.76	76.85	**77.39**	77.18
	sv_lines	84.93	85.07	85.43	86.03	86.05	85.15	85.30	**86.19**	**86.19**
	sv_talbanken	83.85	84.30	85.32	85.85	85.90	85.18	85.41	**86.64**	86.14
old	grc_proiel	75.57	77.37	77.26	**77.46**	76.91	76.76	76.70	76.30	76.47
	grc_perseus*	60.48	**65.13**	64.86	64.17	64.35	64.52	64.41	63.93	64.29
	got_proiel*	72.74	80.36	**80.75**	79.87	79.91	78.97	78.72	78.72	79.50
	la_ittb	90.08	89.98	90.17	89.93	89.62	**90.30**	90.13	90.08	90.16
	la_proiel	77.59	79.56	79.76	**79.99**	78.84	79.07	78.96	79.41	79.13
	la_perseus	56.28	65.34	65.63	70.07	69.72	64.71	65.40	**71.56**	70.05
	cu_proiel*	61.24	64.48	**65.22**	64.88	63.91	64.11	64.42	64.17	64.93
pt-gl	gl_ctg	81.16	80.92	80.94	**81.70**	81.38	80.93	80.89	81.50	81.56
	gl_treegal	70.80	65.73	75.01	81.07	82.11	64.60	68.93	**82.49**	82.28
	pt_bosque	90.45	90.41	90.43	90.52	**90.62**	90.49	90.28	90.44	90.09
sw-sla	hr_set	88.20	88.33	88.43	88.39	88.17	88.35	88.73	**88.84**	88.74
	sr_set	87.20	87.78	88.89	88.94	88.88	88.31	89.16	89.47	**89.48**
	sl_ssj	93.22	**93.54**	93.29	93.39	93.44	93.37	93.30	93.40	93.29
	sl_sst	60.30	70.67	70.78	70.40	70.28	70.31	70.84	**71.22**	71.09
turkic	bxr_bdt*	19.51	26.83	31.71	**36.59**	21.95	26.83	21.95	29.27	31.71
	kk_ktb	14.02	**62.62**	58.88	48.60	49.53	59.81	61.68	51.40	49.53
	tr_imst	65.57	66.02	65.66	64.48	64.86	**66.29**	66.03	65.99	66.00
	ug_udt*	47.85	49.11	49.01	49.18	48.68	**50.39**	49.86	50.06	49.91
w-sla	cs_cac	91.86	92.24	92.19	92.20	92.18	92.12	92.22	92.09	**92.40**
	cs_fictree	92.97	94.01	94.23	94.23	94.41	94.11	94.35	**94.52**	94.10
	cs_pdt	89.57	90.39	90.57	90.45	90.42	90.32	90.34	90.89	**90.89**
	pl_lfg	95.41	93.30	96.17	**96.49**	96.40	92.87	96.19	96.43	96.43
	pl_pdb	91.64	91.29	91.82	91.72	91.72	91.74	91.85	91.72	**91.94**
	sk_snk	92.07	93.62	93.51	**93.77**	93.58	93.13	93.41	93.29	93.02
	hsb_ufal*	14.47	59.21	60.53	63.16	59.21	61.84	**65.79**	61.84	63.16
avg.		78.52	80.63	81.58	**82.16**	81.77	80.60	81.26	**82.10**	81.88

Table 1: LAS scores for each dataset (dev) for all of our settings, both when training a parser per cluster ("Trained on cluster"), as well as having one parser for all treebanks ("Trained on all"). Bold: highest score for this training setup, omitted if the MONO baseline performs best. * not used in mBERT pretraining.

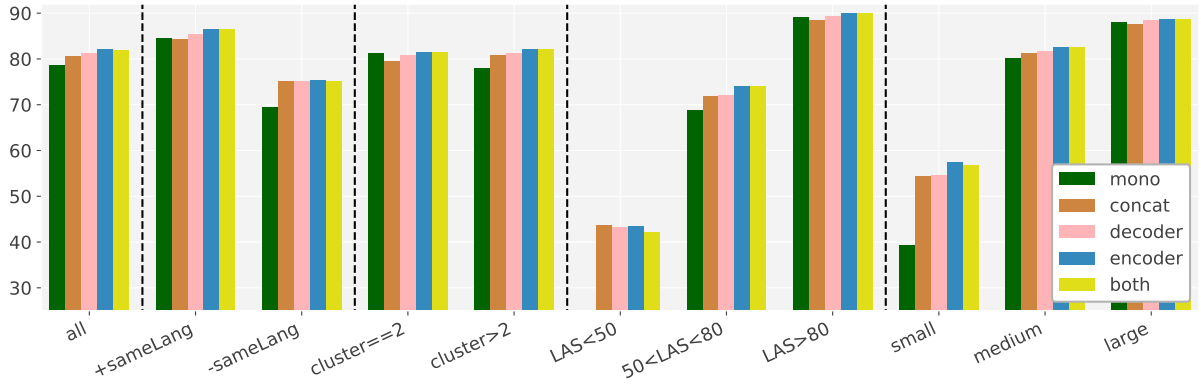

Figure 3: Scores for same subsets as Figure 2, but when training a single parser on all data at once, instead of separate parsers for each cluster.

same language is included in the cluster, and `-sameLang` for all datasets for which this is not the case. `cluster==2`, are scores for clusters consisting of 2 datasets, and `cluster>2` for larger clusters. We also divide the datasets based on their performance with the MONO baseline (LAS<50, 50<LAS<80, LAS>80), and finally based on the size of the training data: small (<1,000 sentences), medium (<10,000) and large (>20,000).

Dataset embeddings are especially beneficial for datasets where performance of the MONO baseline is low (`LAS<50`) and for small datasets. They help moderately for medium sized datasets, and for datasets where performance of MONO is mediocre (`50<LAS<80`). For larger and high-performing datasets, performance increases diminish. The CONCAT baseline outperforms the MONO baseline in most setups, except for small clusters indicating that some of the gains from using the dataset embedding methods in these settings are due to the additional use of data.

Perhaps a bit counterintuively, the dataset embeddings methods improve results more for treebanks for which there is not a treebank of the same language (`-sameLang`). However, this may very well be due to a confounding factor: the scores are generally a lot higher in the `+sameLang` setting than in the `-sameLang` setting and the method seems to work better for treebanks for which the baseline scores are lower.

Overall, the ENCODER model performs best; it either outperforms all others, or performs on par with the best setup. The ENCODER model outperforms the other models mostly on datasets where the MONO baseline is low, and for small datasets. The DECODER strategy is only beneficial in some of the data subsets, and should be used with caution. Perhaps surprisingly, the BOTH strategy is not beneficial over the ENCODER, indicating that both strategies encode dataset information differently.

In Figure 3, we report the results when training one parser on all the treebanks for each setup. Results are very similar to results with the parsers trained per cluster (Figure 2). The main difference can be observed for the datasets with low performance of the MONO baseline (LAS<50), where the difference between the different dataset embeddings and CONCAT is smaller. The similarity to results obtained with smaller clusters indicates that 1) using dataset embeddings is also viable in a highly multilingual model and 2) a highly multilingual model might be able to pick up on dataset similarities and use the relevant data for individual languages. This has practical implications: it can be more practical to have one model that works on many languages (Kondratyuk and Straka, 2019) than multiple models and it removes the need to carefully construct clusters of related languages, which can be time-consuming without a guarantee for optimal clusters.

4.3 Test data

To avoid overusing the test data, we only confirm our main findings on the test data; we compare our best baseline to our best dataset embedding setup for both the setup trained in clusters and when training on all datasets simultaneously. Altough van der Goot (2021) suggests to concatenate the tune set to the

Clusters		All	
CONCAT	ENCODER	CONCAT	ENCODER
81.08	**82.05**	80.57	**82.28**

Table 2: Average LAS scores on the test data from our best baseline (CONCAT), and the best setup with dataset embeddings (ENCODER) for both the cluster trained-parser and the parser trained on all data.

training data for the final comparison on test, we only use the training split to save compute and because we are not aiming for a new state-of-the-art.

Results (Table 2) show that for parsers trained on clusters the CONCAT baseline performs a bit higher compared to the development data (Table 1), and the gain is thus smaller, but still substantial. When training on all datasets, the results are similar. Overall, the results on the test data confirm the findings on the development data: 1) dataset embeddings are useful in both setups, 2) one large multilingual model trained on all treebanks performs on par with multiple parsers trained on clusters of related treebanks.

5 Conclusion

We evaluated the usefulness of dataset embeddings for multilingual parsing using large pretrained multilingual LMs and found them to be useful in this context. Using this method improves over both a monolingual baseline and a baseline where training treebanks are concatenated, and across many settings. This method helps mostly for small treebanks. Using dataset embeddings in the encoder showed overall slightly better results than our other embedding strategies, even better than combining the two approaches to use dataset embeddings. Finally, we found that using dataset embeddings in a multilingual parser that uses training data from all available treebanks works just as well as using them with clusters of treebanks from related languages.

Acknowledgements

We would like to thank the anonymous reviewers, Ahmet Üstün, Max Müller-Eberstein and Daniel Varab for their discussions about dataset embeddings and evaluation. Miryam de Lhoneux was funded by the Swedish Research Council (Grant 2020-00437).

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *Transactions of the Association for Computational Linguistics*, 4:431–444.

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.

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.

Timothy Dozat and Christopher Manning. 2017. Deep Biaffine Attention for Neural Dependency Parsing. In *Proceedings of the 5th International Conference on Learning Representations.*

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Process-*

ing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2779–2795, Hong Kong, China. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018. 82 treebanks, 34 models: Universal Dependency parsing with multi-treebank models. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 113–123, Brussels, Belgium. Association for Computational Linguistics.

Sara Stymne, Miryam de Lhoneux, Aaron Smith, and Joakim Nivre. 2018. Parser training with heterogeneous treebanks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 619–625, Melbourne, Australia. Association for Computational Linguistics.

Rob van der Goot. 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. Association for Computational Linguistics.

Rob van der Goot, Ahmet Üstün, Alan Ramponi, Ibrahim Sharaf, and Barbara Plank. 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 Computational Linguistics: System Demonstrations*, pages 176–197, Online. Association for Computational Linguistics.

David Vilares, Carlos Gómez-Rodríguez, and Miguel A. Alonso. 2016. One model, two languages: training bilingual parsers with harmonized treebanks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 425–431, Berlin, Germany. Association for Computational Linguistics.

Joachim Wagner, James Barry, and Jennifer Foster. 2020. Treebank embedding vectors for out-of-domain dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8812–8818, Online. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. 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)*, pages 833–844, Hong Kong, China. Association for Computational Linguistics.

Daniel Zeman, Jan Hajič, Martin Popel, Martin Potthast, Milan Straka, Filip Ginter, Joakim Nivre, and Slav Petrov. 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. Association for Computational Linguistics.

A Exact Scores

filter	mono	concat	decoder	encoder	both	ALLconc.	ALLdec.	ALLenc.	ALLboth
Avg.-58	78.50	80.71	81.45	81.96	81.82	80.57	81.20	**82.03**	81.94
hasSameLang-35	84.44	84.36	85.62	86.36	86.34	84.17	85.26	**86.48**	86.48
noSameLang-23	69.46	75.17	75.11	75.26	74.95	75.09	75.03	**75.26**	75.03
cluster=2-10	81.15	79.55	80.56	**82.31**	82.11	79.52	80.71	81.51	81.33
cluster>2-48	77.94	80.96	81.64	81.89	81.76	80.79	81.30	**82.14**	82.07
low-5	22.66	43.36	42.57	43.54	42.23	**43.69**	43.14	43.38	42.20
medium-14	68.68	72.26	72.97	73.87	73.84	71.77	72.04	74.00	**74.04**
high-39	89.18	88.54	89.48	89.79	89.77	88.46	89.37	89.87	**89.87**
small-7	39.26	54.45	55.15	57.35	56.45	54.37	54.55	**57.48**	56.71
medium-27	80.15	81.37	81.90	82.36	82.32	81.18	81.70	82.49	**82.50**
large-24	88.09	87.64	88.62	**88.68**	88.66	87.52	88.42	88.67	88.67

Table 3: Exact numbers for results in Figure 2. Average LAS scores over different subsets of the data. All: all data, +sameLang: datasets for which another in-language treebank exists, cluster==2: clusters of size 2, LAS<50: treebanks for which the 'mono´ baseline scores <50 LAS, small, medium, large: datasets with a maximum size of respectively: 1,000, 10,000, 20,000 sentences.