We Need to Talk About train-dev-test Splits

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

Abstract

Standard train-dev-test splits used to benchmark multiple models against each other are ubiquitously used in Natural Language Processing (NLP). In this setup, the train data is used for training the model, the development set for evaluating different versions of the proposed model(s) during development, and the test set to confirm the answers to the main research question(s). However, the introduction of neural networks in NLP has led to a different use of these standard splits; the development set is now often used for model selection during the training procedure.Because of this, comparing multiple versions of the same model during development leads to overestimation on the development data. As an effect, people have started to compare an increasing amount of models on the test data, leading to faster overfitting and "expiration" of our test sets. We propose to use a *tune-set* when developing neural network methods, which can be used for model picking so that comparing the different versions of a new model can safely be done on the development data.[1]

1 Dataset Splits in NLP

1.1 Current State

In Natural Language Processing (NLP), a highly empirical field, it is common to benchmark multiple models to each other on a standard dataset. However, since most current models are supervised, and thus require labeled training data, the datasets have to be split. To ensure a fair comparison, most datasets in NLP have standard splits. Most datasets consist of three splits (also visualized in Figure 1(a)):

- **train**: Used for training models, in some setups this split can be omitted (zero-shot or unsupervised learning).

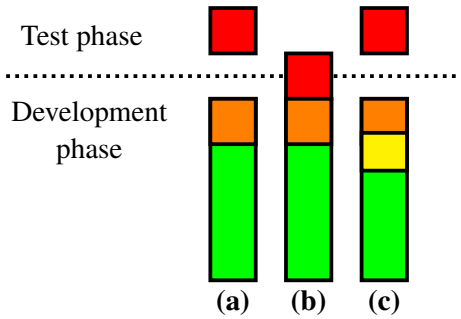

Figure 1: Overview of the use of data splits. red :test orange :dev green :train yellow :tune. a) standard splits for traditional machine learning models b) standard splits as used for neural network models c) our proposed splits for neural network models.

- **dev**elopment (also called validation/evaluation): Used to compare all different versions of the proposed model(s). Can also be used to get preliminary answers to the main research questions.

- **test**: Used to confirm the final answer to the research question.

One often raised worry is that if too many papers are written based on the same test-set, overfitting occurs, especially when only positive results are published (Scargle, 2000). It should be noted that we do not refer to overfitting of the models parameters, but on design decisions (hyperparameters etc.), in line with "bias from research design" as defined by Hovy and Prabhumoye (2021). This means that there is a bias towards methods that perform well on this specific set. We agree that this is a danger. If we consider a more general perspective to this problem, a certain split becomes more prone to this when more different models are evaluated on this exact same data. Let's assume that there is a threshold N that limits the number of times we can re-use the same split for evaluation. The number of papers that can use the same dataset for

[1]Source code is available at https://bitbucket.org/robvanderg/tuneset

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 4485–4494
November 7–11, 2021.

a fair comparison is then equal to N divided by the average number of evaluated models per paper. From this, it follows that, no matter how large N is, a larger average number of runs per paper will drastically reduce the lifespan of a dataset.

For this reason, it used to be common to evaluate all varieties of a newly proposed model on the development data, and only confirm the main findings (e.g. comparison of 2 most relevant models) on the test set. This means that if we propose a new model B, and we want to prove that it outperforms existing model A, we would first evaluate and tune all our varieties of model B ($B_{1...n}$) on the dev set, and then only compare the best version of model B to model A. These varieties of model B can include differences in hyperparameters as well as design decisions. From this it also logically follows that (qualitative) analysis should not be done on the test data.

It should be noted that in some situations a hidden test-set is enforced to circumvent overfitting, for example in shared tasks, where the test data is only shared at the end, and on benchmark websites, where the test-labels remain hidden for the participants (Kim et al., 2011; Wang et al., 2018; Aguilar et al., 2020; Khanuja et al., 2020). This setup is enforced for good reasons, and, in our opinion, should be the standard setup in NLP

1.2 What Has Changed?

Since the introduction of neural networks, the use of the dev set has changed. Neural network models are commonly trained for multiple epochs over the training data, because they are prone to overfitting (on the training data) it is common to evaluate the model on the dev data after each epoch, and use the model from the epoch with the highest performance on the dev data. This "best model selection" (i.e. the best epoch) differs from other hyperparameters, as it is re-tuned every run. In other words, the development data is integrated into the training procedure. This model selection has shown to be important for final performance (Chen and Ritter, 2020). A problem now arises when we want to compare our new model B to model A and train multiple models $B_{1...n}$ on the same dev split. Namely, the performance on the development data of each model B_i is likely to be overly optimistic.

People have noticed this problem, and started to compare multiple versions of their proposed model

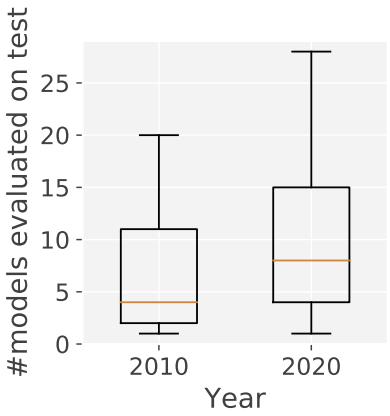

Figure 2: A boxplot visualizing the median and quantiles of the number of models evaluated on test data for a selection of 100 random papers from the ACL 2010 and ACL 2020 proceedings.

B's on the test data (Figure 1(b)).[2] To confirm this trend, we counted the number of novel models (i.e. non-baseline) evaluated on the test data for 100 random papers of the ACL 2010 and 2020 proceedings. Results show a clear trend: in 2020 there are more models evaluated on the test set per paper (Figure 2). For example, in 2010 50% of the papers evaluated less than 4 models on the test data, in 2020 this was the case for only 25% of the papers. The annotator (with 7 years full-time research experience in NLP) observed that in many cases it is not explicitly reported for results on which split they are based (especially in 2020), but in most cases, this could be derived from comparing the analysis results with the main results or from the repository. Furthermore, the papers in 2010 more often used non-benchmark datasets created specifically for a study, for which running multiple models on the test data is arguably less severe. More details about the annotation are reported in the appendix.

To sum up, when using the development set for model picking, one is left with a choice for model comparison: use the dev data or the test data. If the dev data is used, performance is easily overestimated because the model picking was done on the same set. If the test data is used, overfitting of design decisions will more quickly happen on the test data, and it becomes obsolete faster.

[2]this tweet anecdotally shows how it became standard to have seen test data in non-shared task setups: https://twitter.com/marian_nmt/status/1331728574307438597. Furthermore, the EMNLP 2020 call for papers asked authors to report "Corresponding validation performance for each reported test result": https://2020.emnlp.org/call-for-papers

2 Related Work

Gorman and Bedrick (2019) and Çöltekin (2020) propose to use random data splits instead of the standard splits. In other words, they propose to shuffle the whole dataset multiple times, and extract a train, dev, and test split from each random shuffle. This would avoid overfitting, as "*the use of a single standard split, may result in avoidable Type I error*" (Gorman and Bedrick, 2019). As pointed out by Søgaard et al. (2021), these random splits have another danger. It is good practice to create stratified datasplits based on some attributes (e.g., time, speaker, document etc.). This stratified sampling leads to more realistic performance estimates for real-world situations (as we assume we want to employ our models for new samples, from other time-periods, speakers or documents). The problem now becomes that after shuffling and re-splitting, it is very likely that sentences from, for example, the same document are both in the training data and the test data, which leads to (unrealistically) higher performances in the experiments of both Gorman and Bedrick (2019) and Çöltekin (2020). Therefore, Søgaard et al. (2021) propose other strategies to resplit the data. They show that using biased splits better approximate real-world performance on new samples (i.e. from another dataset) as standard splits, but still lead to a large overestimation of performance. In both of these proposed setups (random and biased splits), the splits that are proposed are still train-dev-test splits. This means that if the same splits are used across different papers, over-estimation on either dev/test would still occur (depending which one is used to compare $B_{1...n}$), and overfitting still occurs. If instead, new splits are generated for each paper, overestimation still happens, and direct comparison between different papers is more complex.

Recently, there has been an increasing interest in other aspects of evaluation of NLP models, including automatic testing of specific abilities (Ribeiro et al., 2020), significance testing (Dror et al., 2018; Sadeqi Azer et al., 2020), effect of random seeds (Reimers and Gurevych, 2018) and reproducibility (Fokkens et al., 2013; Cohen et al., 2018; Wieling et al., 2018; Branco et al., 2020; Belz et al., 2021). We consider all of these problems (including random/biased splits) orthogonal to the problem of overfitting on the test set, as in all of the proposed setups/solutions train-dev-test splits are still used. This is also the case for k-fold cross-validation which is a standard method to combat overfitting, within the k folds, there are still dev-sets on which one will overfit if for each fold, hyperparameter tuning, model-picking and analysis is done on the same data.

3 The Tune Split

The solution we propose to the problem introduced in Section 1.2 follows logically from the observation that we do not have a data split left for comparing the models. We simply introduce an additional data split, which we call the *tune* split (Figure 1(c)). This tune data can be used to pick the best model, thereby leaving the development set out of the training procedure. Then the best model B out of $B_{1...n}$ to compare against model A can be picked based on the development data, and the superiority of model B_i can be confirmed on the test data. This also makes a comparison to traditional machine learning models fairer, as they also do not make use of the dev data during training.

One clear downside of this approach is that there is less data remaining for the other splits. To overcome this, one could also pick the best hyperparameters/settings for model B based on the dev split, while using the tune split for model picking, and then for the final comparison add the tune split to the train split and use the development data for picking the best model. This procedure is the same as it would be in a shared task setup, where the train+dev data can be used however the participants see fit, but the test data remains unseen until the final comparison.

It should be noted that in cross-domain or cross-lingual setups, similar solutions have recently been proposed. In these setups, people commonly use the source dataset dev split for model picking (Keung et al., 2020). To have a pure cross-domain or cross-lingual setup, it is important to not tune on all target domains/languages as you are likely to overestimate performance when no target data is available. Artetxe et al. (2020) therefore argue to only use the dev set of one target language and report test results on other languages. ANother case where a similar solution was sometimes used, is the devtest set in machine translation, which is used at least since the WMT 2006 shared task (Koehn and Monz, 2006). This split is an effect of having many sequential shared task, where new test-data is added every year. In some work, the dev split is used for model-picking and the testdev split is used

as development data. However, to the best of our knowledge, there is no offical use (nor guidelines) on the function of the devtest split. An alternative solution is introduced by Chen and Ritter (2020), who propose methods for picking the best model that do not rely on any labeled data.

4 Case Study

To evaluate the effect of having a separate tune split, we perform a case study in which we fine-tune a transition-based (Nivre, 2008) Bi-LSTM (Graves and Schmidhuber, 2005) parser and a transformer-based (Vaswani et al., 2017) deep biaffine parser (Dozat and Manning, 2017) on the same datasets. We use the Universal Dependencies (UD) 2.8 data (Zeman et al., 2021) as benchmark, and use the UUParser (Smith et al., 2018a) and the MaChAmp (van der Goot et al., 2021) implementations of the corresponding parsers.

4.1 Experimental Setup

We use the datasets selected by Smith et al. (2018b). We concatenate the train and dev set (we omit the test data in these experiments, to avoid over-analyzing it), and resplit the resulting data in 4 splits: the last 3,000 sentences are used for 1,000 sentences respectively for the test, dev, and tune split, and the remaining data is used as training data. We do not shuffle the sentences, as they are chronologically ordered in many cases, resulting in a (somewhat) stratified split, thereby avoiding overestimation of performance because train/test have overlapping sources (as done by Gorman and Bedrick (2019) and Çöltekin (2020)).[3] We consider two finetuning setups:

- train+tune for training, model-picking and hyperparameter tuning on dev (Figure 1(a)).
- train for training, model-picking on tune, hyperparameter tuning on dev (Figure 1(c), our proposed setup). In this setup, we concatenate train and tune for the final evaluation on the test set with the optimal hyperparameters.

For both parsers we make a selection of hyperparameters to tune, and take the default values as starting point. We use no external embeddings for the UUParser, and initialize MaChAmp with mBERT to cover a variety of setups. Hence, a fair comparison can only be made between the setups, and not between the parsers. The exact hyperparameter ranges that were evaluated are reported in the appendix. We perform a grid search for each dataset, and compare the performance on test as well as the number of hyperparameters that have a different optimal value across both setups.

[3]We also provide an alternative splitting method for UD data, for setups where the original test-split is to be used for the final comparison, more details can be found in the appendix.

	MaChAmp			UUParser		
Dataset	Dif	-T	+T	Dif	-T	+T
grc_proiel	2/4	72.28	72.19	2/7	78.17	77.38
ar_padt	1/4	82.11	81.82	0/7	77.60	77.63
en_ewt	1/4	88.89	88.90	1/7	82.64	82.90
fi_tdt	2/4	88.41	87.85	1/7	80.50	80.81
zh_gsd	1/4	83.13	82.66	0/7	69.67	69.27
he_htb	2/4	84.49	84.33	1/7	73.22	73.30
ko_gsd	2/4	81.99	82.32	0/7	77.28	77.15
ru_gsd	2/4	88.51	88.48	1/7	80.14	79.84
sv_talbanken	1/4	82.76	82.89	1/7	71.11	71.40

Table 1: Results (LAS) of tuning with both strategies. Dif reports the number of optimal hyperparameters that differ between the two setups, -T(une) is using dev for model picking as well as hyperparameter-tuning, and +T(une) is our proposed setup.

4.2 Results

Results (Table 1) show that performance of both evaluated setups only have minimal differences on the test data.[4] Even though there are different optimal hyperparameters found for all datasets for MaChAmp and for 6/9 for the UUParser, none of the differences are significant with a paired bootstrap test (10,000 resamples), both with and without Bonferroni correction (Bonferroni, 1936). Hence, the results indicate that for the final performance it is irrelevant which splits to use in this setup. However, when the tune split is used, we can do a much more valuable (qualitative or quantitative) analysis on the development data, which would be less realistic to do when we used dev already for hyperparameter tuning as well as model picking.

5 Conclusion

We have reflected on the default dataset splits used in NLP (and actually also more widely in machine learning) to tune design decisions (architectures, hyperparameters, etc.) of neural network based models, which can easily lead to overfitting on

[4]Performance differences between the development and test split are reported in the appendix.

the test data. This is an effect of the fact that in standard setups, neural networks use the development data during training, and it thus became more common to compare multiple versions of the same model on the test data. The solution to this problem is simple, we need another data split to do model picking, or avoid using the dev set in the training procedure, by learning which model to pick using other heuristics (Chen and Ritter, 2020). We call this split the *tune-split*. The only downside of using a separate tune-split, is that there is less data available for the other splits. This can be circumvented by using train+tune for the final (test-)runs of the model. We evaluated the effect of the tune-split for two common NLP benchmarks by tuning two different types of models. One of them showed to be more robust against the evaluated hyperparameter ranges, whereas the other showed a clear performance improvement when using a tune-split. This proposed solution is orthogonal to other proposed practices for a hygienic experimental setup like significance testing, random splits, and evaluating specific abilities of our models.

Acknowledgements

I would like to thank Barbara Plank, Mike Zhang, Max Müller-Eberstein, Maria Barret, Elisa Bassignana, Marija Stepanovic, Christian Hardmeier, and Antonio Toral for feedback on a draft of this paper and discussions about evaluation of NLP models. Thanks to the anonymous reviewers for their interesting suggestions. This research is supported by an Amazon Faculty Research award.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. LinCE: A centralized benchmark for linguistic code-switching evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1803–1813, Marseille, France. European Language Resources Association.

Mikel Artetxe, Sebastian Ruder, Dani Yogatama, Gorka Labaka, and Eneko Agirre. 2020. A call for more rigor in unsupervised cross-lingual learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7375–7388, Online. Association for Computational Linguistics.

Anya Belz, Shubham Agarwal, Anastasia Shimorina, and Ehud Reiter. 2021. A systematic review of reproducibility research in natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 381–393, Online. Association for Computational Linguistics.

Carlo Bonferroni. 1936. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62.

António Branco, Nicoletta Calzolari, Piek Vossen, Gertjan Van Noord, Dieter van Uytvanck, João Silva, Luís Gomes, André Moreira, and Willem Elbers. 2020. A shared task of a new, collaborative type to foster reproducibility: A first exercise in the area of language science and technology with REPROLANG2020. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5539–5545, Marseille, France. European Language Resources Association.

Yang Chen and Alan Ritter. 2020. Model selection for cross-lingual transfer using a learned scoring function. *arXiv preprint arXiv:2010.06127*.

K. Bretonnel Cohen, Jingbo Xia, Pierre Zweigenbaum, Tiffany Callahan, Orin Hargraves, Foster Goss, Nancy Ide, Aurélie Névéol, Cyril Grouin, and Lawrence E. Hunter. 2018. Three dimensions of reproducibility in natural language processing. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Çağrı Çöltekin. 2020. Verification, reproduction and replication of NLP experiments: a case study on parsing Universal Dependencies. In *Proceedings of the Fourth Workshop on Universal Dependencies (UDW 2020)*, pages 46–56, Barcelona, Spain (Online). Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1691–1701, Sofia, Bulgaria. Association for Computational Linguistics.

Kyle Gorman and Steven Bedrick. 2019. We need to talk about standard splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2786–2791, Florence, Italy. Association for Computational Linguistics.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks*, 18(5-6):602–610.

Dirk Hovy and Shrimai Prabhumoye. 2021. Five sources of bias in natural language processing. *Language and Linguistics Compass*, 15(8).

Phillip Keung, Yichao Lu, Julian Salazar, and Vikas Bhardwaj. 2020. Don't use English dev: On the zero-shot cross-lingual evaluation of contextual embeddings. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 549–554, Online. Association for Computational Linguistics.

Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. GLUECoS: An evaluation benchmark for code-switched NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585, Online. Association for Computational Linguistics.

Jin-Dong Kim, Yue Wang, Toshihisa Takagi, and Akinori Yonezawa. 2011. Overview of Genia event task in BioNLP shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 7–15, Portland, Oregon, USA. Association for Computational Linguistics.

Philipp Koehn and Christof Monz, editors. 2006. *Proceedings on the Workshop on Statistical Machine Translation*. Association for Computational Linguistics, New York City.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Nils Reimers and Iryna Gurevych. 2018. Why comparing single performance scores does not allow to draw conclusions about machine learning approaches. *arXiv preprint arXiv:1803.09578*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Erfan Sadeqi Azer, Daniel Khashabi, Ashish Sabharwal, and Dan Roth. 2020. Not all claims are created equal: Choosing the right statistical approach to assess hypotheses. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5715–5725, Online. Association for Computational Linguistics.

Jeffrey D Scargle. 2000. Publication bias: The "file-drawer" problem in scientific inference. *Journal of Scientific Exploration*, 14(1):91–106.

Aaron Smith, Bernd Bohnet, Miryam de Lhoneux, Joakim Nivre, Yan Shao, and Sara Stymne. 2018a. 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.

Aaron Smith, Miryam de Lhoneux, Sara Stymne, and Joakim Nivre. 2018b. 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. Association for Computational Linguistics.

Anders Søgaard, Sebastian Ebert, Jasmijn Bastings, and Katja Filippova. 2021. We need to talk about random splits. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1823–1832, Online. 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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 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. Association for Computational Linguistics.

Martijn Wieling, Josine Rawee, and Gertjan van Noord. 2018. Squib: Reproducibility in computational linguistics: Are we willing to share? *Computational Linguistics*, 44(4):641–649.

Changlong Yu, Jialong Han, Haisong Zhang, and Wilfred Ng. 2020. Hypernymy detection for low-resource languages via meta learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3656, Online. Association for Computational Linguistics.

Daniel Zeman, Joakim Nivre, Mitchell Abrams, Elia Ackermann, Noëmi Aepli, Hamid Aghaei, Željko Agić, Amir Ahmadi, Lars Ahrenberg, Chika Kennedy Ajede, Gabrielė Aleksandravičiūtė, Ika Alfina, Lene Antonsen, Katya Aplonova, Angelina Aquino, Carolina Aragon, Maria Jesus Aranzabe, Bilge Nas Arıcan, Hórunn Arnardóttir, Gashaw Arutie, Jessica Naraiswari Arwidarasti, Masayuki Asahara, Deniz Baran Aslan, Luma Ateyah, Furkan Atmaca, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Keerthana Balasubramani, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, Starkaður Barkarson, Victoria Basmov, Colin Batchelor, John Bauer, Seyyit Talha Bedir, Kepa Bengoetxea, Gözde Berk, Yevgeni Berzak, Irshad Ahmad Bhat, Riyaz Ahmad Bhat, Erica Biagetti, Eckhard Bick, Agnė Bielinskienė, Kristín Bjarnadóttir, Rogier Blokland, Victoria Bobicev, Loïc Boizou, Emanuel Borges Völker, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Adriane Boyd, Anouck Braggaar, Kristina Brokaitė, Aljoscha Burchardt, Marie Candito, Bernard Caron, Gauthier Caron, Lauren Cassidy, Tatiana Cavalcanti, Gülşen Cebiroğlu Eryiğit, Flavio Massimiliano Cecchini, Giuseppe G. A. Celano, Slavomír Čéplö, Neslihan Cesur, Savas Cetin, Özlem Çetinoğlu, Fabricio Chalub, Shweta Chauhan, Ethan Chi, Taishi Chika, Yongseok Cho, Jinho Choi, Jayeol Chun, Alessandra T. Cignarella, Silvie Cinková, Aurélie Collomb, Çağrı Çöltekin, Miriam Connor, Marine Courtin, Mihaela Cristescu, Philemon. Daniel, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Mehmet Oguz Derin, Elvis de Souza, Arantza Diaz de Ilarraza, Carly Dickerson, Arawinda Dinakaramani, Elisa Di Nuovo, Bamba Dione, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Hanne Eckhoff, Sandra Eiche, Marhaba Eli, Ali Elkahky, Binyam Ephrem, Olga Erina, Tomaž Erjavec, Aline Etienne, Wograine Evelyn, Sidney Facundes, Richárd Farkas, Marília Fernanda, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Kazunori Fujita, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Sebastian Garza, Fabrício Ferraz Gerardi, Kim Gerdes, Filip Ginter, Gustavo Godoy, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta González Saavedra, Bernadeta Griciūtė, Matias Grioni, Loïc Grobol, Normunds Grūzītis, Bruno Guillaume, Céline Guillot-Barbance, Tunga Güngör, Nizar Habash, Hinrik Hafsteinsson, Jan Hajič, Jan Hajič jr., Mika Hämäläinen, Linh Hà Mỹ, Na-Rae Han, Muhammad Yudistira Hanifmuti, Sam Hardwick, Kim Harris, Dag Haug, Johannes Heinecke, Oliver Hellwig, Felix Hennig, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Eva Huber, Jena Hwang, Takumi Ikeda, Anton Karl Ingason, Radu Ion, Elena Irimia, Ọlájídé Ishola, Kaoru Ito, Tomáš Jelínek, Apoorva Jha, Anders Johannsen, Hildur Jónsdóttir, Fredrik Jørgensen, Markus Juutinen, Sarveswaran K, Hüner Kaşıkara, Andre Kaasen, Nadezhda Kabaeva, Sylvain Kahane, Hiroshi Kanayama, Jenna Kanerva, Neslihan Kara, Boris Katz, Tolga Kayadelen, Jessica Kenney, Václava Kettnerová, Jesse Kirchner, Elena Klementieva, Arne Köhn, Abdullatif Köksal, Kamil Kopacewicz, Timo Korkiakangas, Natalia Kotsyba, Jolanta Kovalevskaitė, Simon Krek, Parameswari Krishnamurthy, Oğuzhan Kuyrukçu, Aslı Kuzgun, Sookyoung Kwak, Veronika Laippala, Lucia Lam, Lorenzo Lambertino, Tatiana Lando, Septina Dian Larasati, Alexei Lavrentiev, John Lee, Phuong Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Maria Levina, Cheuk Ying Li, Josie Li, Keying Li, Yuan Li, KyungTae Lim, Bruna Lima Padovani, Krister Lindén, Nikola Ljubešić, Olga Loginova, Andry Luthfi, Mikko Luukko, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher Manning, Ruli Manurung, Büşra Marşan, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Hiroshi Matsuda, Yuji Matsumoto, Alessandro Mazzei, Ryan McDonald, Sarah McGuinness, Gustavo Mendonça, Niko Miekka, Karina Mischenkova, Margarita Misirpashayeva, Anna Missilä, Cătălin Mititelu, Maria Mitrofan, Yusuke Miyao, AmirHossein Mojiri Foroushani, Judit Molnár, Amirsaeid Moloodi, Simonetta Montemagni, Amir More, Laura Moreno Romero, Giovanni Moretti, Keiko Sophie Mori, Shinsuke Mori, Tomohiko Morioka, Shigeki Moro, Bjartur Mortensen, Bohdan Moskalevskyi, Kadri Muischnek, Robert Munro, Yugo Murawaki, Kaili Müürisep, Pinkey Nainwani, Mariam Nakhlé, Juan Ignacio Navarro Horñiacek, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Manuela Nevaci, Luong Nguyễn Thị, Huyền Nguyễn Thị Minh, Yoshihiro Nikaido, Vitaly Nikolaev, Rattima Nitisaroj, Alireza Nourian, Hanna Nurmi, Stina Ojala, Atul Kr. Ojha, Adéday Olúòkun, Mai Omura, Emeka Onwuegbuzia, Petya Osenova, Robert Östling, Lilja Øvrelid, Şaziye Betül Özateş, Merve Özçelik, Arzucan Özgür, Balkız Öztürk Başaran, Hyunji Hayley Park, Niko Partanen, Elena Pascual, Marco Passarotti, Agnieszka Patejuk, Guilherme Paulino-Passos, Angelika Peljak-Łapińska, Siyao Peng, Cenel-Augusto Perez, Natalia Perkova, Guy Perrier, Slav Petrov, Daria Petrova, Jason Phelan, Jussi Piitulainen, Tommi A Pirinen, Emily Pitler, Barbara Plank, Thierry Poibeau, Larisa Ponomareva, Martin Popel, Lauma Pretkalniņa, Sophie Prévost, Prokopis Prokopidis, Adam Przepiórkowski, Tiina Puolakainen, Sampo Pyysalo, Peng Qi, Andriela Rääbis, Alexandre Rademaker, Taraka Rama, Loganathan Ramasamy, Carlos Ramisch, Fam Rashel, Mohammad Sadegh Rasooli, Vinit Ravishankar, Livy Real, Petru Rebeja, Siva Reddy, Georg Rehm, Ivan Riabov, Michael Rießler, Erika Rimkutė, Larissa Rinaldi, Laura Rituma, Luisa Rocha, Eiríkur Rögnvaldsson, Mykhailo Romanenko, Rudolf Rosa, Valentin Roșca, Davide Rovati, Olga Rudina, Jack Rueter, Kristján Rúnarsson, Shoval Sadde, Pegah Safari, Benoît Sagot, Aleksi Sahala, Shadi Saleh, Alessio Salomoni, Tanja

Samardžić, Stephanie Samson, Manuela Sanguinetti, Ezgi Sanıyar, Dage Särg, Baiba Saulīte, Yanin Sawanakunanon, Shefali Saxena, Kevin Scannell, Salvatore Scarlata, Nathan Schneider, Sebastian Schuster, Lane Schwartz, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Hiroyuki Shirasu, Yana Shishkina, Muh Shohibussirri, Dmitry Sichinava, Janine Siewert, Einar Freyr Sigurðsson, Aline Silveira, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Maria Skachedubova, Aaron Smith, Isabela Soares-Bastos, Carolyn Spadine, Rachele Sprugnoli, Steinþór Steingrímsson, Antonio Stella, Milan Straka, Emmett Strickland, Jana Strnadová, Alane Suhr, Yogi Lesmana Sulestio, Umut Sulubacak, Shingo Suzuki, Zsolt Szántó, Dima Taji, Yuta Takahashi, Fabio Tamburini, Mary Ann C. Tan, Takaaki Tanaka, Samson Tella, Isabelle Tellier, Marinella Testori, Guillaume Thomas, Liisi Torga, Marsida Toska, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Utku Türk, Francis Tyers, Sumire Uematsu, Roman Untilov, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Andrius Utka, Sowmya Vajjala, Rob van der Goot, Martine Vanhove, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Natalia Vlasova, Aya Wakasa, Joel C. Wallenberg, Lars Wallin, Abigail Walsh, Jing Xian Wang, Jonathan North Washington, Maximilan Wendt, Paul Widmer, Seyi Williams, Mats Wirén, Christian Wittern, Tsegay Woldemariam, Tak-sum Wong, Alina Wróblewska, Mary Yako, Kayo Yamashita, Naoki Yamazaki, Chunxiao Yan, Koichi Yasuoka, Marat M. Yavrumyan, Arife Betül Yenice, Olcay Taner Yıldız, Zhuoran Yu, Zdeněk Žabokrtský, Shorouq Zahra, Amir Zeldes, Hanzhi Zhu, Anna Zhuravleva, and Rayan Ziane. 2021. Universal dependencies 2.8.1. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

A Details of Annotation

The annotator is asked to find the number of runs per test split of each dataset, where the number of reported metrics is not important, and multiple random seeds are not counted as multiple runs. If a paper contains multiple datasets, the "main" dataset is counted, or an average is used. If a figure contains more then 10 versions of a model, it is counted as only 10 different models. Results reported in an appendix are not counted. Baselines (as defined by the original authors) are not counted. We skip papers that do not introduce results of a new model, as well as analysis only papers (a total of 15 papers in 2010 and 5 papers in 2020 are skipped). It should be noted that the annotator observed that analysis papers are often using the test-set for analysis which is undesirable in our opinion, as this easily leads to overfitting. One paper included in this analysis already included a tune-set in a cross-lingual setup (Yu et al., 2020), similar as the papers mentioned in Sectionr̃efsec:relWork.

B Proposed Splits for UD data

We propose a strategy for resplitting the Universal Dependencies (Zeman et al., 2021) datasets so that they also include a tune-set. First, for datasets without development set, we create one from the last 100 sentences of the training data.[5]. Then we take one third of the development split, and use this as the tune split. With this splitting strategy, the training data size remains the same as the original, and we assume that for tuning only little data is necessary (hence we keep a larger dev set).

Code to generate these splits can be found in `scripts/9.udResplit.py` in the repo.

C Experimental details

For our experiments we selected the ranges of hyperparameters reported in Table 2.

D Difference in Performance Between Dev and Test

The difference in score between dev and test for each setup are shown in Table 3. However, it should be noted that this comparison is not completely fair, as for the proposed setup, the final model is trained on more data when getting the scores on the test data, so the difference is expected to be more positive (the difference is significant for both the UUParser and MaChAmp with a paired bootstrap test p=0.05; dataset results are used as samples).

[5]In UD, a training set is prioritized over a dev set if the dataset is small.

MaChAmp	
Learning rate	[**1e-4**, 1e-5]
Dropout	[.2, **.3**, .4]
cut_frac	[.1, **.2**, .3]
decay	[.35, **38**, 5]
UUparser	
Graphs based	[True, **False**]
Learning rate	[1e-2, **1e-3**, 1e-4]
Word emb. size	[50, **100**, 200]
Char emb. size	[100, **500**]
Number BiLSTM layers	[1, **2**]

Table 2: Evaluated hyperparameters of MaChAmp and the UUParser(defaults are bold).

	MaChAmp		UUParser	
	-T	+T	-T	+T
grc_proiel	0.53	1.06	0.99	1.49
ar_padt	0.09	1.25	0.00	2.16
en_ewt	-1.94	-1.77	-0.64	0.47
fi_tdt	-0.94	-1.05	-0.56	0.79
zh_gsd	-0.38	1.70	-0.85	2.86
he_htb	0.99	1.65	0.80	3.29
ko_gsd	-1.80	0.33	-2.27	0.11
ru_gsd	0.01	1.71	-1.08	2.01
sv_talbanken	-3.69	-0.47	-4.91	-0.48

Table 3: Difference in performance between dev and test set. Lower scores indicate that performance on the test set is lower as compared to dev.

E Results of Hyperparameter Search

The optimal hyperparameters for both setups are shown in Table 4 for the UUParser and in Table 5 for MaChAmp.

Data	Graph	LR	WordS.	CharS.	#-layers
UD_Ancient_Greek-PROIEL					
-Tune	1	0.001	100	500	2
+Tune	1	0.0001	50	500	2
UD_Arabic-PADT					
-Tune	1	0.001	50	100	2
+Tune	1	0.001	50	100	2
UD_English-EWT					
-Tune	1	0.001	200	500	2
+Tune	1	0.001	200	100	2
UD_Finnish-TDT					
-Tune	1	0.001	100	100	2
+Tune	1	0.001	50	100	2
UD_Chinese-GSD					
-Tune	1	0.001	50	500	2
+Tune	1	0.001	50	500	2
UD_Hebrew-HTB					
-Tune	1	0.001	50	500	2
+Tune	1	0.001	100	500	2
UD_Korean-GSD					
-Tune	1	0.001	50	500	2
+Tune	1	0.001	50	500	2
UD_Russian-GSD					
-Tune	1	0.001	50	500	2
+Tune	1	0.001	200	500	2
UD_Swedish-Talbanken					
-Tune	1	0.001	100	500	2
+Tune	1	0.001	50	500	2

Table 4: Optimal hyperparameter for the UUParser, both without a tune set (-Tune) and with a tune set (+Tune). WordS. = size of word embeddings, CharS. = size of character embeddings, #layers = number of BiLSTM layers.

Data	LR	dropout	cut_frac	decay
UD_Ancient_Greek-PROIEL				
-Tune	0.0001	0.4	0.1	0.35
+Tune	0.0001	0.4	0.2	0.38
UD_Arabic-PADT				
-Tune	0.0001	0.3	0.2	0.5
+Tune	0.0001	0.4	0.2	0.5
UD_English-EWT				
-Tune	0.0001	0.4	0.2	0.38
+Tune	0.0001	0.4	0.2	0.35
UD_Finnish-TDT				
-Tune	0.0001	0.3	0.2	0.5
+Tune	0.0001	0.4	0.2	0.35
UD_Chinese-GSD				
-Tune	0.0001	0.2	0.1	0.5
+Tune	0.0001	0.3	0.1	0.5
UD_Hebrew-HTB				
-Tune	0.0001	0.3	0.1	0.5
+Tune	0.0001	0.3	0.2	0.38
UD_Korean-GSD				
-Tune	0.0001	0.3	0.2	0.38
+Tune	0.0001	0.3	0.1	0.5
UD_Russian-GSD				
-Tune	0.0001	0.4	0.2	0.5
+Tune	0.0001	0.2	0.3	0.5
UD_Swedish-Talbanken				
-Tune	0.0001	0.3	0.2	0.5
+Tune	0.0001	0.4	0.2	0.5

Table 5: Optimal hyperparameter for MaChAmp, both without a tune set (-Tune) and with a tune set (+Tune).