Towards Realistic Practices In Low-Resource Natural Language Processing: The Development Set

Katharina Kann, Kyunghyun Cho and Samuel R. Bowman
New York University, USA
{kann, kyunghyun.cho, bowman}@nyu.edu

Abstract

Development sets are impractical to obtain for real low-resource languages, since using all available data for training is often more effective. However, development sets are widely used in research papers that purport to deal with low-resource natural language processing (NLP). Here, we aim to answer the following questions: Does using a development set for early stopping in the low-resource setting influence results as compared to a more realistic alternative, where the number of training epochs is tuned on development *languages*? And does it lead to overestimation or underestimation of performance? We repeat multiple experiments from recent work on neural models for low-resource NLP and compare results for models obtained by training with and without development sets. On average over languages, absolute accuracy differs by up to 1.4%. However, for some languages and tasks, differences are as big as 18.0% accuracy. Our results highlight the importance of realistic experimental setups in the publication of low-resource NLP research results.

1 Introduction

Parametric machine learning models are frequently trained by minimizing the loss on the training set $\mathcal{T}$,

$$\mathcal{L}_{\mathcal{T}}(\boldsymbol{\theta}) = \sum_{x \in \mathcal{T}} l\left(\theta, x\right), \tag{1}$$

for model parameters θ and a predefined loss function l. Gradient-based optimizers minimize this loss $\mathcal{L}(\boldsymbol{\theta})$ by updating θ in the direction of the gradient $\nabla\mathcal{L}(\boldsymbol{\theta})$. A low loss characterizes a model which makes accurate predictions for examples in $\mathcal{T}$. However, since $\mathcal{T}$ is finite, overfitting the training set might lead to poor generalization performance. One way to avoid fitting Equation 1 too closely is *early stopping*: a separate *development* or *validation* set is used to end training as soon as the loss on the development set $\mathcal{L}_{\mathcal{D}}(\boldsymbol{\theta})$ starts increasing or model performance on the development set $\mathcal{D}$ starts decreasing. The best set of parameters $\boldsymbol{\theta}$ is used in the final model.

	# train	# dev	ES
Bollmann et al. (2018)	5k	12k-46k	Yes
Kann et al. (2018)	400-700	100-200	Yes
Makarov and Clematide (2018)	100	1k	Yes
Sharma et al. (2018)	100	100	Yes
Schulz et al. (2018)	1k-21k	9k	N/A
Upadhyay et al. (2018)	500	1k	Yes

Table 1: Number of examples used for training and development in recent low-resource NLP experiments; *ES*=early stopping on the development set. Experiments from papers in bold will be revisited here.

This works well when large amounts of data are available to create training, development and test splits. Recently, however, with the success of pretraining (Peters et al., 2018; Devlin et al., 2019) and multi-task learning (Caruana, 1997; Ruder, 2017; Wang et al., 2019) approaches, neural models are showing promising results on various natural language processing (NLP) tasks also in *low-resource* or *few-shot* settings (Johnson et al., 2017; Kann et al., 2017; Yu et al., 2018). Often, the high-resource experimental setup and training procedure are kept unchanged, and the size of the original training set is reduced to simulate limited data. This leads to settings where validation examples may outnumber training examples. Table 1 shows such cases for the tasks of historical text normalization (Bollmann et al., 2018), morphological segmentation (Kann et al., 2018), morphological inflection (Makarov and Clematide, 2018; Sharma et al., 2018), argument component identification (Schulz et al., 2018), and transliteration (Upadhyay et al., 2018).

However, in a real-world setting with limited resources, it is unlikely that such a development set would be available for early stopping, since it would be more effective to use at least part of it for training instead. Here, we investigate how previous results relate to those obtained in a setting that does *not* assume a development set. Instead of early stopping, we use data from the same task in other languages, the *development languages*, to decide on the number of training epochs. We are interested in two questions: Does recent work in low-resource NLP overestimate model performance by using an unrealistically precise performance signal to stop training? Or, inversely, is model performance underestimated by overfitting the finite development set?

Our experiments on historical text normalization, morphological inflection, and transliteration, featuring a variety of languages, show that performance does differ between runs with and without early stopping on the development set; if using the development set leads to better or worse results depends on the task and language. Differences of up to 18% absolute accuracy highlight that a realistic evaluation of models for low-resource NLP is crucial for estimating real-world performance.

2 Related Work

Realistic evaluation of machine learning. Oliver et al. (2018) investigate how to evaluate semi-supervised training algorithms in a realistic way; they differ from us in that they focus exclusively on semi-supervised learning (SSL) algorithms, and do not consider NLP explicitly. However, in line with our conclusion, they report that recent practices for evaluating SSL techniques do not address the question of the algorithms' real-word applicability in a satisfying way. In NLP, several earlier works have explicitly investigated real-world low-resource settings as opposed to artificial proxy settings, e.g., for part-of-speech tagging (Garrette et al., 2013) or machine translation (Irvine and Callison-Burch, 2013). While those mostly focus on real data-poor languages, we explicitly investigate the effect of the common practice to assume a relatively large development set for early stopping in the low-resource setting.

Low-resource settings in NLP. Research in the area of neural methods for low-resource NLP has gained popularity in recent years, with a dedicated workshop on the topic appearing in 2018 (Haffari et al., 2018). High-level key words under which other work on neural networks for data-poor scenarios in NLP can be found are domain adaptation (Daume III, 2007), multi-task learning (Caruana, 1997; Ruder, 2017), few-shot/zero-shot/one-shot learning (Johnson et al., 2017; Finn et al., 2017), transfer learning (Yarowsky et al., 2001), semi-supervised training (Zhu, 2005), or pretraining (Erhan et al., 2010).

While options for early stopping without a development set exist (Mahsereci et al., 2017), they require hyperparameter tuning, which might not be feasible without a development set, and, most importantly, they are not commonly used in low-resource NLP research. Here, we investigate if *current practices* might lead to unrealistic results.

3 Experimental Design

We compare early stopping using development set accuracy (*DevSet*) with an alternative strategy where the amount of training epochs is a hyperparameter tuned on development languages (*DevLang*). We perform two rounds of training:

- **Stopping point selection phase.** Models for the development languages are trained with the original early stopping strategy from previous work. The number of training epochs for the target languages is then calculated as the average over the best epochs for all development languages.[1] All development languages also function as target languages. To make this possible, for development languages, we compute the average over *other* development languages only.

- **Main training phase.** We train models for all languages keeping both the model resulting from the original early stopping strategy (DevSet) and that from the epoch computed in the stopping point selection phase (DevLang).[2]

The stopping point selection phase exclusively serves the purpose of tuning the number of epochs for the DevLang training setup. Models obtained

[1]We round this number to an integer. It is important for our experiments that the training sets for all languages are of the same size, cf. Table 1. Otherwise we would need to account for the number of training examples per epoch.

[2]This requires training for at least the number of target epochs, even if early stopping would end training earlier.

in this phase are discarded. The development sets we use in our experiments are those from the original papers without alterations.

Since both final models obtained in the main training phase result from the same training run, our experimental design enables a direct comparison between the models from both setups.

Example. Assume that, in the stopping point selection phase, we obtain the best development set results for a given task in development languages L1 and L2 after epochs 14 and 18, respectively. In the main training phase, we then train a model for the same task in target language L3 with the original early stopping strategy, but keeping additionally the model from epoch 16. If the best development result for language L3 is obtained after epoch 19, we compare the model from epoch 19 (DevSet) to that from epoch 16 (DevLang).

4 Tasks, Data, Models

For our study, we select previously published experiments which fulfill the following criteria: (1) datasets exist for at least four languages, and all training sets are of equal size; (2) the original authors use early stopping with a development set; (3) the authors explicitly investigate low-resource settings; and (4) the original code is publically available, or a standard model is used. Since our main goal is to confirm the effect of the development set and not to compare between tasks, we further limit this study to sequence-to-sequence tasks.

4.1 Historical Text Normalization (NORM)

Task. The goal of historical text normalization is to convert old texts into a form that conforms with contemporary spelling conventions. Historical text normalization is a specific case of the general task of text normalization, which additionally encompasses, e.g., correction of spelling mistakes or normalization of social media text.

Data. We experiment on the ten datasets from Bollmann et al. (2018), which represent eight different languages: German (two datasets; Bollmann et al., 2017; Odebrecht et al., 2017); English, Hungarian, Icelandic, and Swedish (Pettersson, 2016); Slovene (two datasets; Ljubešic et al., 2016); and Spanish and Portuguese (Vaamonde, 2015). We treat the two datasets for German and Slovene as different languages. All languages serve both as development languages for all *other* languages and as target languages.

Model. Our model for this task is an LSTM (Hochreiter and Schmidhuber, 1997) encoder-decoder model with attention (Bahdanau et al., 2015). Both encoder and decoder have a single hidden layer. We use the default model in OpenNMT (Klein et al., 2017)[3] as our implementation and employ the hyperparameters from Bollmann et al. (2018). In the original paper, *early stopping* is done by training for 50 epochs, and the best model regarding development accuracy is applied to the test set.

4.2 Morphological Inflection (MORPH)

Task. Morphological inflection consists of mapping the canonical form of a word, the lemma, to an indicated inflected form. This task gets very complex for morphologically rich languages, where a single lemma can have hundreds or thousand of inflected forms. Recently, morphological inflection has frequently been cast as a sequence-to-sequence task, mapping the characters of the input word together with the morphological features specifying the target to the characters of the corresponding inflected form (Cotterell et al., 2018).

Data. We experiment on the datasets released for a 2018 shared task (Cotterell et al., 2018), which cover 103 languages and feature an explicit low-resource setting. We randomly choose ten development languages: Armenian, Basque, Galician, Georgian, Greenlandic, Icelandic, Karbadian, Kannada, Latin, and Lithuanian.

Model. For MORPH, we experiment with a pointer-generator network architecture (Gu et al., 2016; See et al., 2017). This is a sequence-to-sequence model similar to that for NORM, but employs separate encoders for characters and features. It is further equipped with a copy mechanism: using attention to decide on what element from the input sequence to copy, the model computes a probability for either copying or generation while producing an output. The final probability distribution over the target vocabulary is a combination of both. Hyperparameters are taken from Sharma et al. (2018).[4] For *early stopping*, we also follow Sharma et al. (2018): all models are trained

[3] `github.com/OpenNMT/OpenNMT-py`
[4] `github.com/abhishek0318/conll-sigmorphon-2018`

for at least 300 epochs, and training is continued for another 100 epochs each time there has been improvement on the development set within the last 100 epochs.

4.3 Transliteration (TRANSL)

Task. Transliteration is the task of converting names from one script into another, while staying as close to the original pronunciation as possible. Unlike for *translation*, focus lies on the sound; the target language meaning is usually ignored.

Data. For our transliteration experiments, we follow Upadhyay et al. (2018). We experiment on datasets from the Named Entities Workshop 2015 (Duan et al., 2015) in Hindi, Kannada, Bengali, Tamil, and Hebrew. For this task, all languages are both development and target languages.

Model. The last featured model is an LSTM sequence-to-sequence model similar to that by Bahdanau et al. (2015), except for using hard monotonic attention (Aharoni and Goldberg, 2017). It attends to a *single* character at a time, and attention moves monotonically over the input. We take hyperparameters and code from Upadhyay et al. (2018).[5] *Early stopping* is done by training for 20 epochs and applying the best model regarding development accuracy to the test data.

4.4 Experimental Setup

We run all experiments using the implementations from previous work or OpenNMT as described above. Existing code is only modified where necessary. Most importantly, we add storing of the DevLang model during the main training phase.

5 Results

Development sets vs. development languages. We are asking if the use of a development set for early stopping leads to over- or underestimation of realistic model performance. Thus, we show in Table 2 how often we obtain higher accuracy for each of DevLang and DevSet. Additionally, averaged performance over all languages as well as the maximum difference in absolute accuracy are listed in Table 3. For NORM and TRANSL, results for individual languages are shown in Tables 4 and 5, respectively; for detailed results for MORPH see Appendix A. We see in Table 2 that, for MORPH and NORM, the use of unrealistically large development sets leads to better results than DevLang for 72 and 8 languages, respectively. For 8 and, respectively, 2 languages there is no difference, and a look at the detailed results in Appendix A and Table 4 reveals that, for those cases, we end up training for the same number of epochs for DevLang and DevSet. Only in 23 cases for MORPH, and none for NORM, we obtain *better* results for DevLang. This suggests that, for these two tasks, we frequently overestimate realistic model performance by early stopping on the development set. Indeed, Table 3 confirms this finding and shows that, on average across languages, DevSet models outperform DevLang models. The maximum difference is 18% absolute accuracy for the language Azeri and MORPH: for DevSet, we reach 64% accuracy after epoch 217, while, for DevLang, we only obtain 46% accuracy after the predefined epoch 324.

We obtain a different picture for TRANSL: results are equal for 3 languages, and better for DevLang for the remaining 2. Equal performance might be explained by the overall smaller number of training epochs in the original regime: stopping at the same epoch for both strategies is more likely. Overall, for TRANSL, performance on the development set seems to be less predictive for the final test performance than for the other tasks.

Influence of the final epoch. Since without a development set performance decreases on MORPH for most languages, we investigate if this can be explained by training often being too short. Therefore, we plot the difference of training du-

	MORPH	NORM	TRANSL
DevLang>DevSet	23	0	2
DevLang=DevSet	8	2	3
DevLang<DevSet	72	8	0

Table 2: Summary of cases in which using a development set leads to overestimation (DevSet>DevLang), underestimation (DevSet<DevLang), or neither (DevSet=DevLang) of the final model performance.

	MORPH	NORM	TRANSL
DevSet	**51.3**	**74.9**	21.8
DevLang	50.0	74.2	**22.3**
Δ	-1.4	-0.7	+0.5
max Δ	-18.0	-2.4	+1.3

Table 3: Test accuracy in % for different stopping approaches and tasks, averaged over languages.

[5] github.com/shyamupa/hma-translit

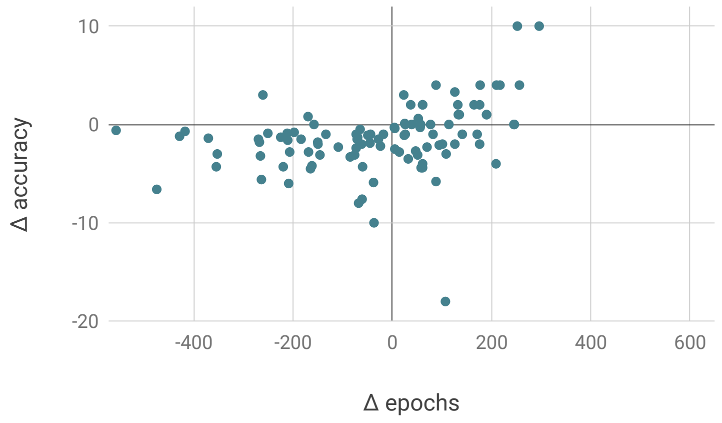

Figure 1: Difference in accuracy (DevLang-DevSet) depending on the difference in training epochs (DevLang-DevSet) for MORPH.

Language	DevSet	DevLang
English	0.7705 (26)	0.7682 (43.44)
German (1)	0.6749 (44)	0.6749 (44.44)
German (2)	0.7075 (43)	0.6837 (41.78)
Hungarian	0.4897 (44)	0.4773 (41.67)
Icelandic	0.7017 (27)	0.6952 (43.56)
Portuguese	0.7944 (43)	0.7860 (42.44)
Slovene (1)	0.8206 (38)	0.8202 (41.67)
Slovene (2)	0.8952 (46)	0.8873 (42.00)
Spanish	0.8352 (42)	0.8352 (41.89)
Swedish	0.7985 (43)	0.7913 (42.11)

Table 4: Detailed results per language for NORM; corresponding epochs in parenthesis.

Language	DevSet	DevLang
Bengali	0.374 (9)	0.387 (10.50)
Hebrew	0.135 (10)	0.135 (10.00)
Hindi	0.248 (10)	0.248 (10.25)
Kannada	0.216 (11)	0.229 (10.00)
Tamil	0.116 (10)	0.116 (10.25)

Table 5: Detailed results per language for TRANSL; corresponding epochs in parenthesis.

ration in epochs between DevLang and DevSet against the resulting difference in accuracy in Figure 1. While we indeed find that shorter training duration for a pointer-generator network for this task mostly results in worse performance, longer training can lead to either lower or higher accuracy. Thus, while longer training seems better for MORPH, not all performance loss can be explained by aborting training too early.

6 Discussion And Conclusion

Limitations. We investigate the effect of early stopping on the validation set as compared to a realistic setting without target language development examples. However, we would like to point out that, in certain situations, standard practices might be sufficient, e.g., for comparing different methods in equal settings, if absolute performance is not the main focus.

Further, we do not claim to show that using a validation set always *over*- or *under*estimates real-world performance, since this depends on how representative the validation set is of the target distribution. Our main result is that using a development set gives a poor estimate of real-world performance and that it is important to be aware of potential performance differences.

Practical take-aways. We replicate experiments from recent low-resource NLP research, once with the original experimental design and once without assuming development sets for early stopping. Since differences in absolute accuracy are up to 18.0%, we conclude that low-resource NLP research should move away from using large development sets for early stopping whenever real-world settings are being considered.

Acknowledgments

We would like to thank Nikita Nangia and Jason Phang for their feedback on this work. This work has benefited from the support of Samsung Research under the project *Improving Deep Learning using Latent Structure* and from the donation of a Titan V GPU by NVIDIA Corporation.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Marcel Bollmann, Joachim Bingel, and Anders Søgaard. 2017. Learning attention for historical text normalization by learning to pronounce. In *ACL*.

Marcel Bollmann, Anders Søgaard, and Joachim Bingel. 2018. Multi-task learning for historical text normalization: Size matters. In *DeepLo*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *CoNLL–SIGMORPHON*.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Xiangyu Duan, Rafael E. Banchs, Min Zhang, Haizhou Li, and A Kumaran. 2015. Proceedings of the fifth named entity workshop. In *Fifth Named Entity Workshop*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *JMLR*, 11:625–660.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of POS-taggers for low-resource languages. In *ACL*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Reza Haffari, Colin Cherry, George Foster, Shahram Khadivi, and Bahar Salehi. 2018. Proceedings of the workshop on deep learning approaches for low-resource NLP.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ann Irvine and Chris Callison-Burch. 2013. Combining bilingual and comparable corpora for low resource machine translation. In *WMT*.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Googles multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. In *ACL*.

Katharina Kann, Jesus Manuel Mager Hois, Ivan Vladimir Meza Ruiz, and Hinrich Schütze. 2018. Fortification of neural morphological segmentation models for polysynthetic minimal-resource languages. In *NAACL*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *ACL*.

Nikola Ljubešic, Katja Zupan, Darja Fišer, and Tomaz Erjavec. 2016. Normalising slovene data: historical texts vs. user-generated content. In *KONVENS*.

Maren Mahsereci, Lukas Balles, Christoph Lassner, and Philipp Hennig. 2017. Early stopping without a validation set. *arXiv:1703.09580*.

Peter Makarov and Simon Clematide. 2018. Imitation learning for neural morphological string transduction. In *EMNLP*.

Carolin Odebrecht, Malte Belz, Amir Zeldes, Anke Lüdeling, and Thomas Krause. 2017. RIDGES herbology: designing a diachronic multi-layer corpus. *Language Resources and Evaluation*, 51(3):695–725.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. 2018. Realistic evaluation of deep semi-supervised learning algorithms. In *NeurIPS*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

Eva Pettersson. 2016. *Spelling normalisation and linguistic analysis of historical text for information extraction*. Ph.D. thesis, Acta Universitatis Upsaliensis.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv:1706.05098*.

Claudia Schulz, Steffen Eger, Johannes Daxenberger, Tobias Kahse, and Iryna Gurevych. 2018. Multi-task learning for argumentation mining in low-resource settings. In *NAAACL*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.

Abhishek Sharma, Ganesh Katrapati, and Dipti Misra Sharma. 2018. IIT(BHU)–IIITH at CoNLL–SIGMORPHON 2018 shared task on universal morphological reinflection. In *CoNLL–SIGMORPHON*.

Shyam Upadhyay, Jordan Kodner, and Dan Roth. 2018. Bootstrapping transliteration with constrained discovery for low-resource languages. In *EMNLP*.

Gael Vaamonde. 2015. Userguide for digital edition of texts in ps post scriptum.

Alex Wang, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *ICLR*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT*.

Mo Yu, Xiaoxiao Guo, Jinfeng Yi, Shiyu Chang, Saloni Potdar, Yu Cheng, Gerald Tesauro, Haoyu Wang, and Bowen Zhou. 2018. Diverse few-shot text classification with multiple metrics. *arXiv:1805.07513*.

Xiaojin Jerry Zhu. 2005. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, Department of Computer Sciences.

A Detailed Results for MORPH

Language	DevSet	DevLang
adyghe	89.2 (263)	84.8 (324)
albanian	27.9 (400)	24.8 (324)
arabic	34.1 (397)	31.7 (324)
armenian	52.2 (393)	50.9 (323.1111111)
asturian	71.2 (348)	69 (324)
azeri	64 (217)	46 (324)
bashkir	80.1 (474)	78.1 (324)
basque	7.9 (282)	8.5 (334.2222222)
belarusian	22.1 (198)	25.4 (324)
bengali	56 (585)	59 (324)
breton	56 (263)	58 (324)
bulgarian	50.3 (266)	45.9 (324)
catalan	64 (319)	61.5 (324)
classical-syriac	93 (285)	93 (324)
cornish	32 (267)	32 (324)
crimean-tatar	90 (458)	89 (324)
czech	37.2 (592)	35.4 (324)
danish	64.5 (409)	61.2 (324)
dutch	54.1 (362)	48.2 (324)
english	87.3 (292)	83.8 (324)
estonian	31 (536)	30.1 (324)
faroese	34 (310)	31.2 (324)
finnish	21.5 (799)	14.9 (324)
french	55.1 (881)	54.5 (324)
friulian	72 (247)	72 (324)
galician	44 (230)	41.9 (324.4444444)
georgian	77.2 (353)	76.1 (303.5555556)
german	52.3 (384)	48 (324)
greek	25.6 (300)	24.5 (324)
greenlandic	74 (167)	76 (331.6666667)
haida	58 (298)	57 (324)
hebrew	31.3 (268)	31 (324)
hindi	75 (594)	73.5 (324)
hungarian	39.5 (395)	38 (324)
icelandic	33.9 (250)	31.6 (319.8888889)
ingrian	50 (263)	46 (324)
irish	23.7 (493)	20.9 (324)
italian	45.6 (549)	44.3 (324)
kabardian	89 (233)	86 (341.4444444)
kannada	52 (93)	56 (349.2222222)
karelian	88 (147)	92 (324)
kashubian	52 (263)	54 (324)
kazakh	82 (287)	84 (324)
khakas	82 (148)	80 (324)
khaling	23.1 (742)	22.4 (324)
kurmanji	81.9 (470)	78.8 (324)
ladin	66 (236)	70 (324)
latin	14.4 (478)	10.2 (315.8888889)
latvian	35.3 (397)	34.3 (324)
lithuanian	15.8 (447)	14 (296.5555556)
livonian	31 (134)	32 (324)

Table 6: Detailed results per language for MORPH; corresponding epochs in parenthesis; part 1.

Language	DevSet	DevLang
lower-sorbian	42 (590)	38.8 (324)
macedonian	56.8 (508)	55.3 (324)
maltese	26 (189)	27 (324)
mapudungun	86 (107)	90 (324)
middle-french	82.9 (236)	77.1 (324)
middle-high-german	80 (223)	78 (324)
middle-low-german	36 (192)	38 (324)
murrinhpatha	38 (72)	48 (324)
navajo	10.8 (368)	9.8 (324)
neapolitan	77 (301)	80 (324)
norman	60 (148)	62 (324)
northern-sami	18.7 (695)	17.3 (324)
north-frisian	43 (386)	41 (324)
norwegian-bokmaal	66.3 (273)	63.2 (324)
norwegian-nynorsk	51.6 (385)	44 (324)
occitan	69 (183)	68 (324)
old-armenian	31.6 (389)	31.1 (324)
old-church-slavonic	49 (134)	50 (324)
old-english	22.7 (433)	20.4 (324)
old-french	40.6 (352)	39.1 (324)
old-irish	2 (78)	2 (324)
old-saxon	25 (494)	25.8 (324)
pashto	36 (114)	40 (324)
persian	51.2 (277)	48.5 (324)
polish	31.3 (544)	27 (324)
portuguese	57.3 (319)	56.9 (324)
quechua	58.7 (753)	57.5 (324)
romanian	35.2 (531)	32.4 (324)
russian	43.6 (320)	43.3 (324)
sanskrit	50.7 (489)	46.2 (324)
scottish-gaelic	76 (361)	66 (324)
serbo-croatian	34.4 (533)	28.4 (324)
slovak	41 (575)	40.1 (324)
slovene	47.4 (535)	45.8 (324)
sorani	27.3 (677)	24.3 (324)
spanish	54.3 (588)	48.7 (324)
swahili	62 (298)	62 (324)
swedish	63.7 (392)	55.7 (324)
tatar	76 (191)	77 (324)
telugu	98 (79)	98 (324)
tibetan	36 (28)	46 (324)
turkish	36.6 (299)	36.7 (324)
turkmen	82 (115)	78 (324)
ukrainian	33.8 (522)	33 (324)
urdu	66.6 (369)	64.7 (324)
uzbek	93 (242)	92 (324)
venetian	76.8 (210)	76.8 (324)
votic	22 (342)	21 (324)
welsh	48 (482)	48 (324)
west-frisian	50 (153)	49 (324)
yiddish	63 (198)	61 (324)
zulu	32.3 (679)	28 (324)

Table 7: Detailed results per language for MORPH; corresponding epochs in parenthesis; part 2.