Seq2seq for Morphological Reinflection: When Deep Learning Fails

Hajime Senuma
University of Tokyo
National Institute of Informatics
senuma@nii.ac.jp

Akiko Aizawa
National Institute of Informatics
University of Tokyo
aizawa@nii.ac.jp

Abstract

Recent studies showed that the sequence-to-sequence (seq2seq) model is a promising approach for morphological reinflection. At the CoNLL-SIGMORPHON 2017 Shared Task for universal morphological reinflection, we basically followed the approach with some minor variations. The results were remarkable in a certain sense. In high-resource scenarios our system achieved 91.46% accuracy (only modestly behind the best system by 3.85%), and in medium-resource scenarios the performance was 65.06% (almost the same as baseline). In low-resource settings, however, the performance was only 1.58%, ranking the worst among submitted systems. In this paper, we present system description and error analysis for the results.

1 Introduction

Processing morphological inflection is a fundamental task for the analysis and generation of natural languages and serves as a building block for many tasks such as machine translation, text analytics, and question answering. Whereas English is morphologically simple and abundant for resources, other languages are often morphologically rich and resource-poor, resulting in severe performance degradation (Tsarfaty et al., 2010). To tackle the issue, the CoNLL-SIGMORPHON 2017 Shared Task hosted a shared task on universal morphological reinflection (Cotterell et al., 2017), in which participants must solve the task for 52 languages and for high-, medium-, and low-resource settings.

Although the shared task comprised two subtasks, we participated only in Task 1. Each data set in Task 1 consists of three columns. The first and second column provides a *lemma* and a *target form*, respectively. The third column lists morphosyntactic descriptions (MSDs), or the features for a target form, where each feature is taken from a universal set of morphological features called UniMorph (Sylak-Glassman et al., 2015). The purpose of the task is to construct a system which can estimate a target form from a lemma and its MSDs. For each of 52 languages, participants cope with the problem under varying sizes of training data (10,000 for high, 1,000 for medium, and 100 for low). The use of external resources are not permitted in the main track, but allowed as a separate track.

To solve the problem, we basically followed Kann and Schütze (2016a), the winner of the Shared Task in the previous year (Cotterell et al., 2016). Unfortunately, our approach experienced severe difficulties in low-resource settings. In high-resource settings, our system achieved 91.46% accuracy, the 12th among the 20 systems. In medium-resource setting, the performance was 65.06%, almost the same as that of the baseline (64.7%). And in low-resource setting, the system achieved only 1.58%. The cause of the problem is that if we decrease the number of examples, at some point, the accuracy of deep learning-based approach drastically drops. For our system, the point is somewhere between 110 and 150; at 150, we still retain the accuracy around 36% but at 110, the result becomes nonsensical (see Section 5).

This paper is organized as follows. In Section 2, we briefly summarize related researches in this field. In Section 3, we describe the system description of our approach. In Section 4, we present environmental settings used in our experiments and the main results of our work. In Section 5, we discuss the error analysis of our results.

Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection, pages 100–109,
Vancouver, Canada, August 3–4, 2017.

2 Related Work

Morphological inflection has a long-tradition in natural language processing (NLP). The earliest studies used finite-state transducers (Karttunen, 1983; Koskenniemi, 1984; Kaplan and Kay, 1994). The advantages of the approach are that rules are often hand-crafted and thus suitable in low-resource settings and that it is relatively easy and direct to incorporate the linguistic knowledge of specialists. On the other hand, manual crafting of such rules is often expensive and usually language specialists are not easily available. General purpose open-source libraries for this approach include OpenFST (Allauzen et al., 2007) and Foma (Hulden, 2009). In addition, there are several language-specific systems such as TRMorph for Turkish (Çöltekin, 2010) and HornMorpho for the languages of the Horn of Africa (Gasser, 2011).

In this decade, machine learning for morphological inflection became a hot topic. One direction is to exploit the paradigmatic nature of inflection (Durrett and DeNero, 2013; Ahlberg et al., 2015). For example, Durrett and DeNero (2013) proposed a multi-step supervised learning approach. The first phase tries to extract transformational rules from data sets and consists of three sub-steps: the alignment of words in training data, merging spans across the resulting alignments, and rule extraction from these intermediary information. And then the second phase tries to learn the position and the type of transformation application. The advantage of this approach is that we can obtain concrete paradigms of inflection.

Another recent innovation in this field (Faruqui et al., 2016; Kann and Schütze, 2016b) is the use of the sequence-to-sequence (seq2seq) model (Sutskever et al., 2014) (also known as the encoder-decoder model (Cho et al., 2014)). Notably, Kann and Schütze (2016a) applied the attention-based version of seq2seq models (Bahdanau et al., 2015) to the SIGMORPHON 2016 Shared Task (Cotterell et al., 2016) and showed that their system can learn morphological reinflection even for extremely morphologically rich languages such as Maltese and became the winner of the year.

3 System Description

Our implementation is based on the system of Kann and Schütze (2016a). We will release the implementation under a BSD lincense on the GitHub account of the first author [1].

3.1 Basic architecture

3.1.1 Seq2seq model

Fig. 1 shows the basic archicture of our system. The figure depicts how an input tuple `(dun, V;PST)` is converted to an output string `dunned`.

In its basic form, the seq2seq model consists of two recurrent neural networks (RNNs), the encoder and the decoder. After the encoder is feeded with a sequence of input symbols, the hidden layer of the encoder is used as an input to the decoder, and finally the decoder emits a sequence of output symbols. In reality, RNNs are substituted by gated recurrent units (GRUs), inputs are encoded as bidirectional sequences, and the decoder also gets an attentional information from a context vector (Bahdanau et al., 2015).

Given an input example which consists of a lemma and a set of features, a sequence of symbols for the system is represented as $\{S_{Start} f^+ x^+ S_{End} \mid f \in \Sigma_\phi, x \in \Sigma_L\}$, where S_{Start} and S_{End} represents a start symbol and an ending symbol respectively, Σ_ϕ a set of features, Σ_L a set of symbols in a language, and $+$ the repetition of one or more symbols. To improve the predictive efficiency, f^+ should be sorted by some criteria (Kann and Schütze, 2016a), such as lexicographic order (in Fig. 1, `V;PST` is sorted as $S_{PST} S_V$). Likewise, an output string is encoded as $\{S_{Start} x^+ S_{End} \mid f \in x \in \Sigma_L\}$.

For more details, see Bahdanau et al. (2015) and Kann and Schütze (2016a).

3.1.2 Loss function

Following Faruqui et al. (2016), we used the negative log-likelihood of the output character sequence for our loss function.

3.2 Differences from previous studies

In this section, we describe the differences between our work and previous researches.

3.2.1 Dimension

We used 300 for symbol embeddings, 200 for hidden layers, and 200 for context (attention) vectors, while Kann and Schütze (2016a) used 300 for symbol embeddings and 100 for hidden layers (the dimension of context vectors was not described). We increased the size of hidden layers because at

[1] https://github.com/hajimes/conll2017-system

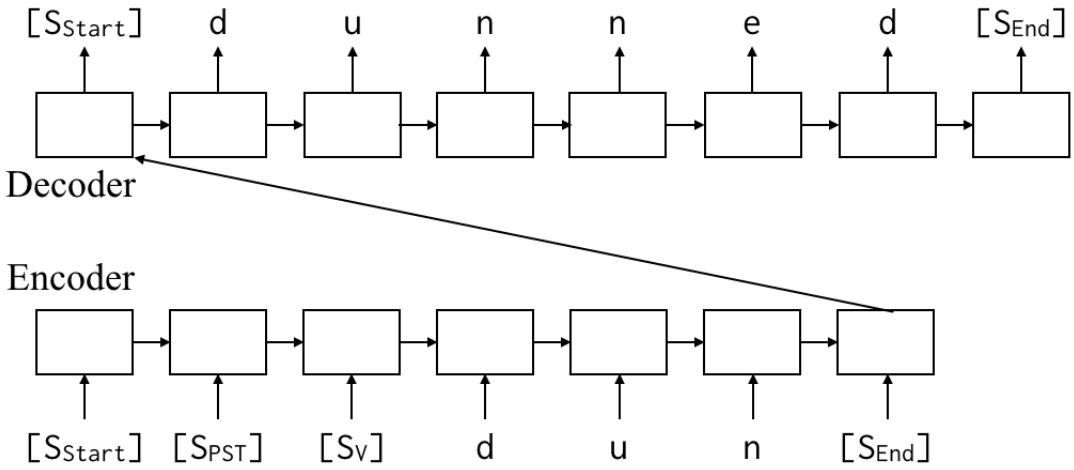

Figure 1: The basic architecture of Kann and Schütze (2016a)'s seq2seq model for morphological reinflection.

least in this task we found that 100 for hidden layers was harmful to predictive performance.

3.2.2 Implementation

We implemented the attention-based version of an encoder-decoder model from scratch with Theano (The Theano Development Team, 2016), while Kann and Schütze (2016a) reused Bahdanau et al. (2015)'s original implementation.

3.2.3 Initialization

We used the Glorot uniform (Glorot and Bengio, 2010) for matrix initialization, while Kann and Schütze (2016a) used the identity matrix.

3.2.4 Optimization / regularization

Faruqui et al. (2016) used AdaDelta (Zeiler, 2012) with L_2 regularization. Kann and Schütze (2016a) also used the same optimizer.

We used the AdaMax optimization algorithm (Kingma and Ba, 2015) with recommended hyperparameters in the paper. The method is a combination of Adam optimization and L_∞ regularization; that is, the bigger the maximum of parameters is, the bigger the penalty for the model is. The reason we used AdaMax is that the method is known for fast convergence. Furthermore, the authors provided recommended hyperparameters, resulting in less hyperparameter calibration.

3.2.5 Iteration number

While Kann and Schütze (2016a) simply used 20 training iterations for any language, we continued training until they are converged: four consecutive no gains in accuracy for development data where the maximum is 40 iterations (for some languages, we hand-tuned the number of training iterations so this number may vary).

4 Experimental Results

4.1 Environmental settings

We used Amazon Web Services (AWS) and ran our system on an Amazon EC2 `p2.16xlarge` instance, Ubuntu with CUDA 8.0 and cuDNN 6.0. The instance was equipped with the eight cards of NVIDIA Tesla K80 (16 GPUs in total).

We trained our model with purely online learning manner (no mini-batch). Although clock-time for training depends on language, under high-resource setting, usually it took about 7 minutes to train a model by using 10,000 examples (that is, one iteration for high-resource training dataset) with one GPU. Hence Time=30 in Table 1 implies training for the language under high-resource setting took about 210 minutes (using one GPU). We only participated in the main track, so we did not use any external resources.

4.2 Results

Table 1 shows the results of our system, descending order of the results for test data set in high-resource setting.

Morphologically simple languages such as English and Persian seem to give high accuracy. Agglutinative languages such as Turkish also tend to contribute to good results. On the other hand, highly-inflectional languages such as Latin give bad performance.

	High				Medium			Low		
Language	Base	Dev	Test	Time	Base	Dev	Test	Base	Dev	Test
norwegian-bokmal	0.750	0.901	0.896	40	0.590	0.772	0.767	0.417	0.015	0.003
georgian	0.933	0.982	0.974	38	0.900	0.864	0.885	0.793	0.012	0.012
lower-sorbian	0.866	0.960	0.953	37	0.670	0.607	0.591	0.362	0.008	0.012
norwegian-nynorsk	0.610	0.866	0.817	37	0.604	0.588	0.557	0.439	0.003	0.007
ukrainian	0.808	0.900	0.908	37	0.734	0.576	0.572	0.523	0.006	0.007
icelandic	0.617	0.850	0.813	36	0.531	0.384	0.412	0.439	0.006	0.006
irish	0.474	0.834	0.831	36	0.424	0.325	0.319	0.317	0.002	0.000
macedonian	0.942	0.936	0.934	36	0.832	0.752	0.762	0.396	0.005	0.002
slovak	0.777	0.942	0.917	36	0.720	0.614	0.614	0.647	0.010	0.006
kurmanji	0.875	0.917	0.920	35	0.790	0.770	0.762	0.633	0.002	0.002
navajo	0.408	0.853	0.851	35	0.385	0.310	0.318	0.306	0.006	0.007
russian	0.900	0.872	0.861	35	0.830	0.526	0.531	0.412	0.000	0.000
serbo-croatian	0.863	0.870	0.888	35	0.570	0.425	0.418	0.285	0.000	0.001
lithuanian	0.662	0.863	0.860	34	0.615	0.404	0.387	0.536	0.003	0.005
slovene	0.798	0.957	0.952	34	0.767	0.629	0.658	0.616	0.013	0.020
faroese	0.651	0.831	0.814	33	0.559	0.365	0.390	0.513	0.003	0.002
northern-sami	0.562	0.928	0.926	32	0.499	0.346	0.344	0.314	0.004	0.005
danish	0.827	0.924	0.884	31	0.753	0.757	0.750	0.567	0.012	0.012
italian	0.901	0.969	0.960	31	0.839	0.866	0.861	0.769	0.001	0.000
bulgarian	0.819	0.951	0.964	30	0.640	0.683	0.655	0.553	0.006	0.004
estonian	0.581	0.965	0.963	30	0.551	0.687	0.658	0.385	0.003	0.001
latin	0.493	0.740	0.709	30	0.449	0.308	0.307	0.336	0.001	0.000
latvian	0.877	0.926	0.922	30	0.852	0.629	0.619	0.790	0.004	0.000
armenian	0.856	0.952	0.941	29	0.785	0.729	0.732	0.722	0.001	0.000
czech	0.841	0.903	0.906	29	0.610	0.620	0.603	0.307	0.003	0.000
polish	0.794	0.885	0.881	29	0.694	0.497	0.496	0.506	0.000	0.002
portuguese	0.975	0.985	0.979	29	0.969	0.827	0.855	0.951	0.007	0.010
sorani	0.646	0.883	0.873	29	0.661	0.565	0.581	0.534	0.007	0.009
english	0.900	0.954	0.942	28	0.832	0.904	0.904	0.784	0.006	0.011
romanian	0.773	0.835	0.828	28	0.630	0.456	0.460	0.151	0.002	0.000
swedish	0.723	0.868	0.875	28	0.635	0.658	0.680	0.421	0.004	0.002
arabic	0.566	0.918	0.902	27	0.553	0.549	0.536	0.380	0.002	0.002
dutch	0.845	0.954	0.943	26	0.796	0.750	0.731	0.588	0.006	0.005
hebrew	0.547	0.984	0.990	26	0.417	0.656	0.673	0.380	0.006	0.006
albanian	0.942	0.985	0.983	25	0.882	0.594	0.612	0.160	0.001	0.003
catalan	0.965	0.967	0.964	25	0.958	0.787	0.772	0.942	0.004	0.004
turkish	0.825	0.940	0.935	25	0.613	0.596	0.607	0.124	0.000	0.001
finnish	0.709	0.845	0.841	24	0.720	0.490	0.512	0.517	0.000	0.000
khaling	0.840	0.996	0.989	24	0.546	0.718	0.693	0.247	0.005	0.009
german	0.705	0.854	0.857	23	0.662	0.594	0.609	0.610	0.004	0.003
quechua	0.972	0.999	0.996	23	0.973	0.914	0.902	0.973	0.008	0.006
hindi	0.961	1.000	1.000	22	0.746	0.839	0.819	0.698	0.009	0.012
persian	0.889	0.994	0.996	21	0.911	0.743	0.717	0.822	0.006	0.005
french	0.982	0.853	0.811	20	0.893	0.713	0.680	0.864	0.003	0.001
spanish	0.954	0.951	0.950	20	0.911	0.798	0.803	0.787	0.001	0.002
urdu	0.991	0.993	0.996	20	0.680	0.875	0.891	0.670	0.045	0.047
welsh	0.752	1.000	0.980	18	0.693	0.850	0.860	0.601	0.010	0.020
hungarian	0.585	0.815	0.791	16	0.453	0.534	0.528	0.255	0.000	0.000
basque	0.060	0.990	1.000	11	0.051	0.750	0.810	0.040	0.020	0.050
haida	0.690	0.990	0.990	9	0.802	0.890	0.880	0.554	0.010	0.010
bengali	0.847	0.990	0.990	4	0.847	0.950	0.950	0.661	0.010	0.010
scottish-gaelic	-	-	-	-	0.441	0.860	0.800	0.449	0.360	0.320

Table 1: Results of our system. Base represents the baseline system provided the organizers. Dev represents the best result for development data. Test represents the final result of our system. Time represents the number of examples for training convergence (unit: 10k). Note that Scottish Gaelic for the high-resource setting is omitted because the data was not provided.

In high-resource setting, our system achieved 91.46% accuracy, the 12th among the 20 systems. In medium-resource setting, the performance was almost the same as baseline 65.06%. And in low-resource setting, the system achieved only 1.58%.

4.3 Comparison with other systems

A heat map in Fig. 2 shows the accuracy of participants under high-resource settings, with the descending order of the average accuracy. Green color (light color in black-and-white) denotes high accuracy whereas red (purple at the extreme) color (dark color in black-and-white) denotes low accuracy. Note that the ranking is slightly different from the official one, because in this figure, if a system did not participate in some languages, we treated them as zero accuracy.

As we see, it is hard to tell the difference, because top systems achieved nearly 100% accuracy for almost all languages. However, if we carefully examined, almost all systems (which have higher performance than the baseline) have similar color spotting patterns, possibly because these participants used similar systems, that is, the seq2seq model (Faruqui et al., 2016; Kann and Schütze, 2016a; Kann et al., 2016). We also see that the color of the Latin language tends to be yellowish or reddish, which indicates that this language is very hard to process by using the seq2seq model.

Heat maps in Fig. 3 and Fig. 4, which depict the case of medium- and low-resource settings, are also interesting to see.

Let us see the case of low-resource settings. It is easy to recognize the systems of `UA` took unique approaches. Other systems have similar color patterns—it may indicate they used the seq2seq model—but the intensity of colors gradually degrades according to the ranking of these systems. Then, after crossing a certain point, the color suddenly becomes purple (nearly 0%) for almost all languages (`EHU-01-0` and our system `UTNII-01-0`).

We will release these figures on the GitHub account of the first author [2].

[2] https://github.com/hajimes/conll2017-stats

5 Discussion

5.1 Convergence speed under high-resource settings

As we see in Table 1, the number of training time for morphological reinflection significantly differs from each language. In the case of Bengali, only 40k (4 iteration for the data set) was sufficient to achieve the best result, whereas Norwegian Bokmål requires 400k (40 iteration). This contrasts with Kann and Schütze (2016a)'s approach where they simply used 20 iterations for any language.

The following table is a list of the top five languages for fast convergence.

Language	Base	Dev	Test	Time
bengali	0.847	0.990	0.990	4
haida	0.690	0.990	0.990	9
basque	0.060	0.990	1.000	11
hungarian	0.585	0.815	0.791	16
welsh	0.752	1.000	0.980	18

On the other hand, training for the following five languages was slow to converge.

Language	Dev	Test	Time
norwegian-bokmal	0.901	0.896	40
georgian	0.982	0.974	38
lower-sorbian	0.960	0.953	37
norwegian-nynorsk	0.866	0.817	37
ukrainian	0.900	0.908	37

5.2 Accuracy under low-resource settings

To test why our system gave catastrophic results under the low-resource setting, we tested more fine-grained analysis as to the size of resources.

We made several new datasets from `english-train-medium` with the size 100, 110, 130, 150, and 500. After we trained our model on these datasets, model-100 gave the best result 0.022 on development data, model-110 0.029, model-130 0.149, model-150 0.356, model-500 0.843, (and model-1000 0.904 as seen in Table 1). It seems that a big trench for our system happens to lie somewhere between 110 and 150 (or 130 and 150)—except Scottish-Gaelic (see Table 1 and Fig. 4). This may explain the reason for big gaps in accuracy with other participants; crossing the trench or not, that is the question. The abrupt decline of predictive performance was also observed by Kann and Schütze (2016b).

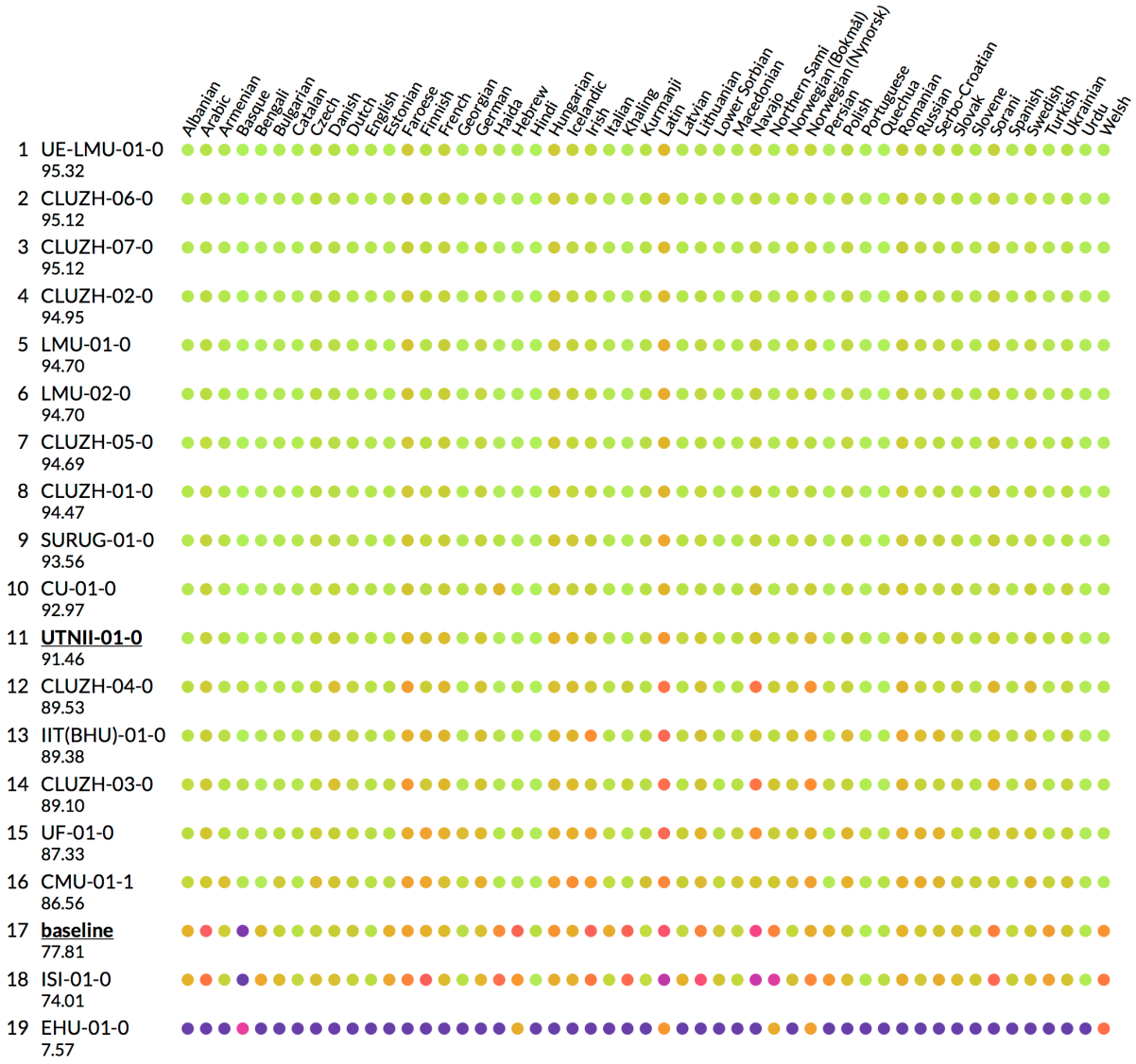

Figure 2: Comparison with other systems under high-resource settings. The signature of our system is `UTNII-01-0`. The ranking is slightly different from the official one, because in this figure, if a system did not participate in some languages, we treated them as zero accuracy. The last number of a system name denotes the usage of external resources (0 = no, 1 = yes).

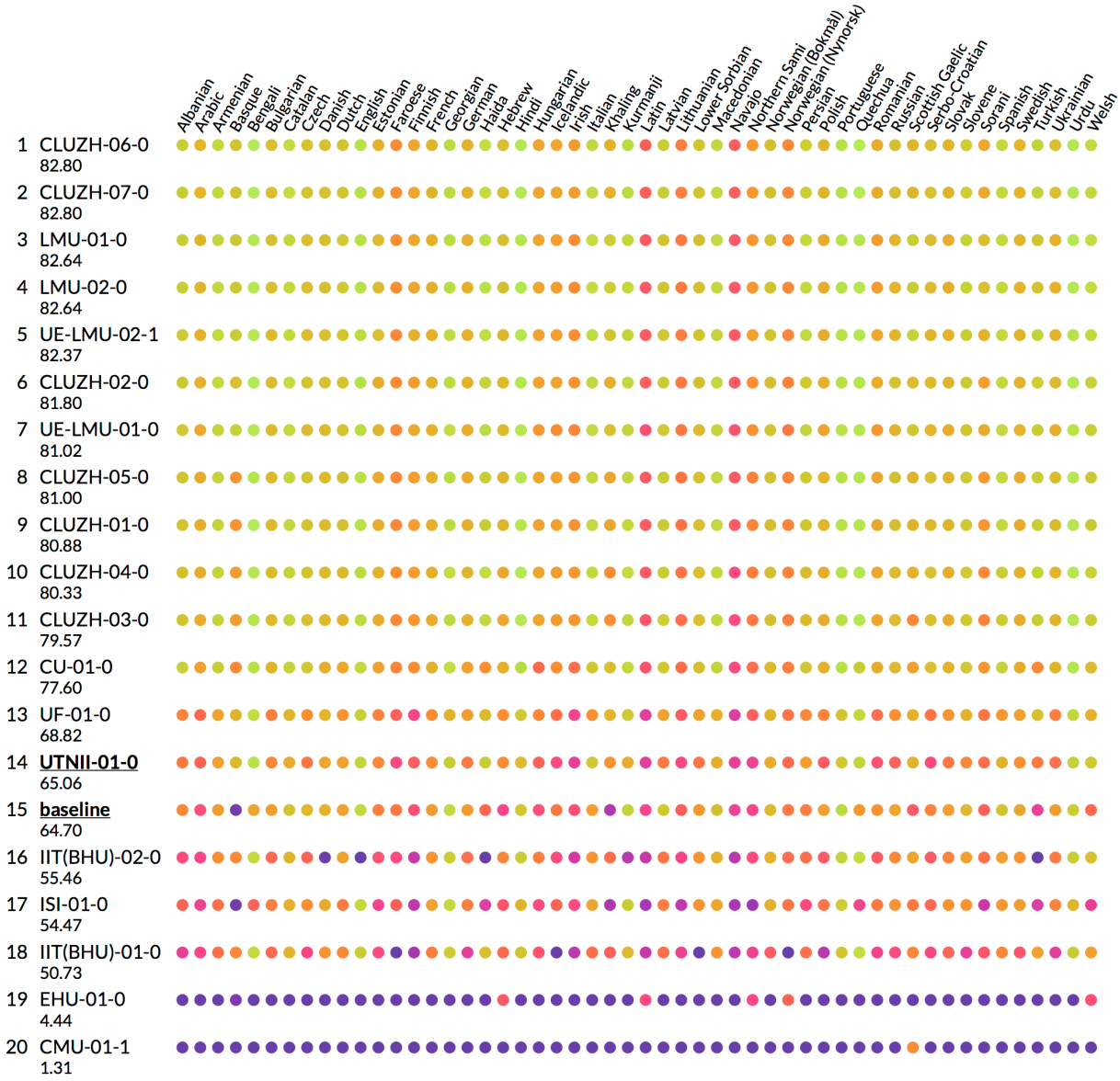

Figure 3: Comparison with other systems under medium-resource settings. The signature of our system is `UTNII-01-0`. The ranking is slightly different from the official one, because in this figure, if a system did not participate in some languages, we treated them as zero accuracy. The last number of a system name denotes the usage of external resources (0 = no, 1 = yes).

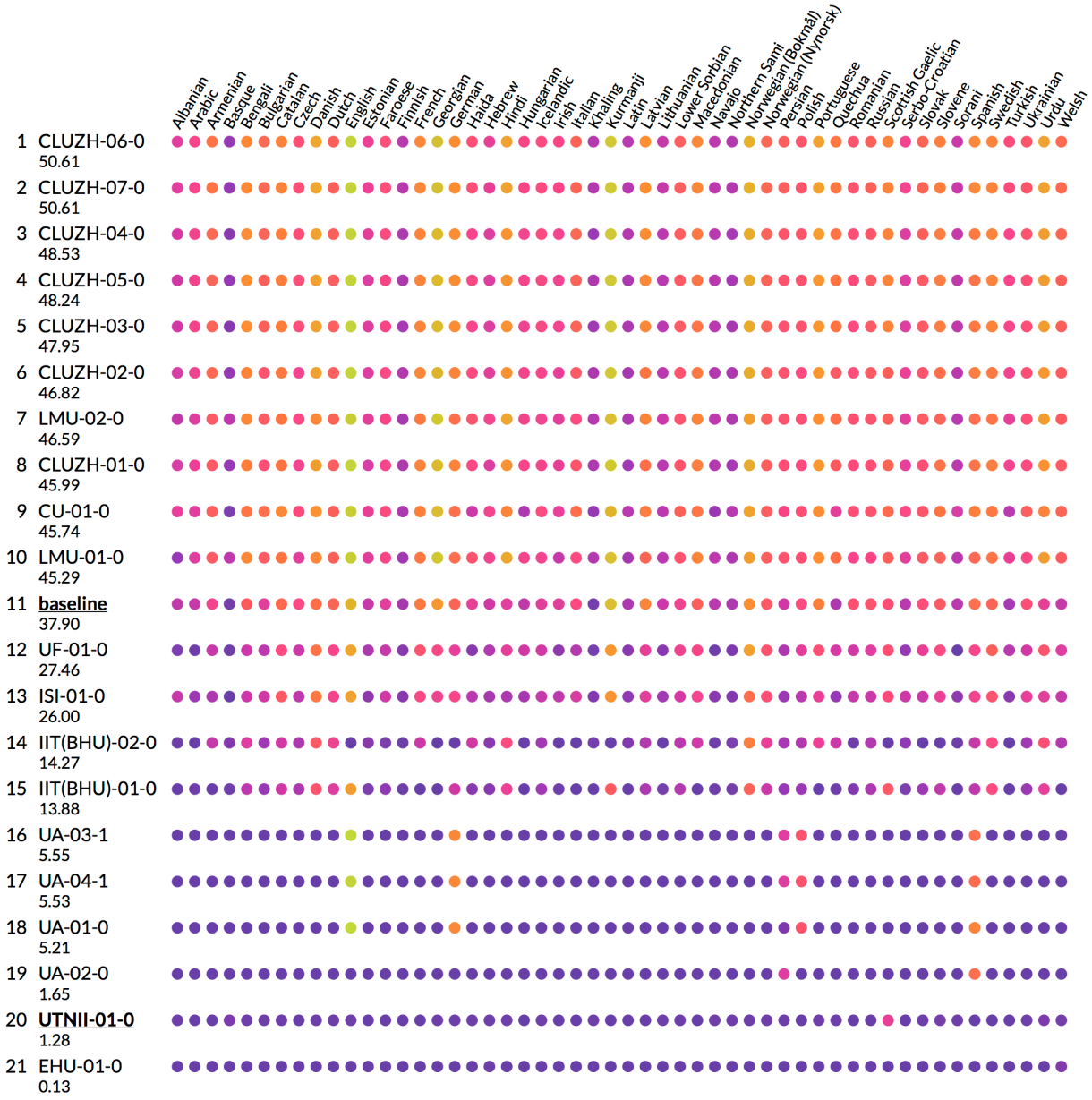

Figure 4: Comparison with other systems under low-resource settings. The signature of our system is `UTNII-01-0`. The ranking is slightly different from the official one, because in this figure, if a system did not participate in some languages, we treated them as zero accuracy. The last number of a system name denotes the usage of external resources (0 = no, 1 = yes).

One possible solution to mitigate this situation is to use other regularization approaches such as dropout (Srivastava et al., 2014), where different configurations are trained simultaneously and probabilistically, although such techniques alone may not change the inherent nature of our system. We will try to find how we can lower the trench in the future.

It is interesting that our system achieved meaningful accuracy for Scottish-Gaelic even under low-resource settings. Although this tendency is not global, the system of `IIT(BHU)-01-0` also shows relatively good performance on the language, so the robustness for processing Scottish-Gaelic may not be by pure chance. We plan to analyze the language in detail, because it will reveal what kind of linguistic natures determine the "trench" of the required number of training examples for seq2seq systems.

6 Conclusion

In this paper, we presented system description and error analysis for our system submitted to the CoNLL-SIGMORPHON 2017 Shared Task. As the reader sees in our results, pure deep learning approaches have a major disadvantage, that is, their predictive performance drops steeply after crossing a certain point of the number of training examples. We also showed that the convergence speed for training the models of morphological reinflection highly depends on the type of languages, which can be useful information to tackle the task again in the future.

Acknowledgements

This work was supported by CREST, Japan Science and Technology Agency. We are also grateful to two anonymous reviewers for their helpful comments.

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 1024–1029. https://doi.org/10.3115/v1/N15-1107.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In *Proceedings of the Twelfth International Conference on Implementation and Application of Automata*. pages 11–23.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation By Jointly Learning To Align and Translate. In *Proceeding of the 3rd International Conference on Learning Representations (ICLR2015)*. http://arxiv.org/abs/1409.0473v3.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN EncoderDecoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 1724–1734. https://doi.org/10.3115/v1/D14-1179.

Çağrı Çöltekin. 2010. A Freely Available Morphological Analyzer for Turkish. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. European Language Resources Association (ELRA), Valletta, Malta, pages 820–827.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. The CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection in 52 Languages. In *Proceedings of the CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 Shared Task—Morphological Reinflection. In *Proceedings of the 14th Annual SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. pages 10–22. https://doi.org/10.18653/v1/W16-2002.

Greg Durrett and John DeNero. 2013. Supervised Learning of Complete Morphological Paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, June, pages 1185–1195.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological Inflection Generation Using Character Sequence to Sequence Learning. In *Proceedings of NAACL-HLT 2016*. pages 634–643. https://doi.org/10.18653/v1/N16-1077.

Michael Gasser. 2011. HornMorpho: a system for morphological processing of Amharic, Oromo, and

Tigrinya. In *Conference on Human Language Technology for Development*. Alexandria, Egypt, pages 94–99.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*. volume 9, pages 249–256.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL 2009*. Association for Computational Linguistics, Athens, Greece, pages 29–32.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2016. Neural Morphological Analysis: Encoding-Decoding Canonical Segments. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 961–967. https://doi.org/10.18653/v1/D16-1097.

Katharina Kann and Hinrich Schütze. 2016a. MED: The LMU System for the SIGMORPHON 2016 Shared Task on Morphological Reinflection. In *Proceedings of the 14th Annual SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. pages 62–70. https://doi.org/10.18653/v1/W16-2010.

Katharina Kann and Hinrich Schütze. 2016b. Single-Model Encoder-Decoder with Explicit Morphological Representation for Reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 555–560. https://doi.org/10.18653/v1/P16-2090.

Ronald M. Kaplan and Martin Kay. 1994. Regular Models of Phonological Rule Systems. *Computational Linguistics* 20(3):331–378.

Lauri Karttunen. 1983. KIMMO: A General Morphological Parser. *Texas Linguistic Forum* 22:165–186.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations 2015*. arxiv:1412.6980v8.

Kimmo Koskenniemi. 1984. A General Computational Model for Word-Form Recognition and Production. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*. pages 178–181. http://aclweb.org/anthology/P84-1038.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15:1929–1958. http://jmlr.org/papers/v15/srivastava14a.html.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. pages 3104–3112.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A Language-Independent Feature Schema for Inflectional Morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 674–680. https://doi.org/10.3115/v1/P15-2111.

The Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions https://arxiv.org/abs/1605.02688.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical Parsing of Morphologically Rich Languages (SPMRL) What, How and Whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*. pages 1–12. http://dl.acm.org/citation.cfm?id=1868772.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. Technical report. http://arxiv.org/abs/1212.5701.