Training Data Augmentation for Low-Resource Morphological Inflection

Toms Bergmanis[*]
t.bergmanis@sms.ed.ac.uk

Katharina Kann[†]
kann@cis.lmu.de

Hinrich Schütze[†]
inquiries@cislmu.org

Sharon Goldwater[*]
sgwater@inf.ed.ac.uk

[*]School of Informatics
University of Edinburgh

[†]Center for Information & Language Processing
LMU Munich

Abstract

This work describes the UoE-LMU submission for the CoNLL-SIGMORPHON 2017 Shared Task on Universal Morphological Reinflection, Subtask 1: given a lemma and target morphological tags, generate the target inflected form. We evaluate several ways to improve performance in the 1000-example setting: three methods to augment the training data with identical input-output pairs (i.e., autoencoding), a heuristic approach to identify likely pairs of inflectional variants from an unlabeled corpus, and a method for cross-lingual knowledge transfer. We find that autoencoding random strings works surprisingly well, outperformed only slightly by autoencoding words from an unlabelled corpus. The random string method also works well in the 10,000-example setting despite not being tuned for it. Among 18 submissions our system takes 1st and 6th place in the 10k and 1k settings, respectively.

1 Introduction

Morphological variation is a major contributor to the sparse data problem in NLP, especially for under-resourced languages. The SIGMORPHON 2016 Shared Task (Cotterell et al., 2016) and CoNLL-SIGMORPHON 2017 Shared Task (Cotterell et al., 2017) aimed to inspire researchers to develop better systems for morphological inflection across a wide range of languages with varying training resources. In 2016, when over 10,000 training examples were provided for each language, neural network-based systems performed considerably better than other approaches (Cotterell et al., 2016). The 2017 Shared Task therefore included settings with different amounts of training data (100, 1000, or 10,000 examples), to examine which approaches might work best when data is even more limited.

We focus on the 1000-example setting of Subtask 1: given a lemma with its part of speech and target morphological tags, generate the target inflected form. Our baseline system is the winning system from 2016, the morphological encoder-decoder (MED) from LMU (Kann and Schütze, 2016). We explore methods for improving its performance in the face of fewer training examples, either with or without additional data from unlabeled corpora.[1]

In particular, we focus on various training data augmentation methods. One uses a heuristic approach to identify likely pairs of inflectional variants from an unlabeled corpus in an unsupervised way, and uses these as input-output pairs. Three other methods augment the training data with identical input-output pairs—i.e., simultaneously training the network to perform autoencoding. We compare three types of autoencoder inputs: either lemmas and inflected forms from the training data, words from an unlabeled corpus, or random strings.

We present detailed results and comparisons for various amounts of added training data for all 52 languages of the shared task. We find that all methods improve considerably over the MED baseline (7.2-10.7% across all languages; a 15.5% improvement over the shared task baseline). Most of the benefit comes with only 4k extra examples, but performance continues to improve up to 16k added examples.

After controlling for the amount of additional data, we see only a small benefit from autoencoding corpus words rather than random strings. Using hypothesized morphological variants works

[1]We did not pursue the 100-example setting because preliminary experiments yielded such low baseline performance that we felt additional work would be unlikely to help much.

Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection, pages 31–39,
Vancouver, Canada, August 3–4, 2017.

as well as random strings. These results suggest that the main advantage of all these methods is providing a strong bias towards learning the identity transformation and/or working as regularizers, rather than learning language-specific phonotactics or morpho-phonological changes.[2]

Finally, following Kann et al. (2017), we test whether cross-lingual knowledge transfer can help, by multilingual training of joint models for groups of up to 10 related languages. However, we find no improvement as compared to using an equivalent amount of random-string autoencoder examples.

Our final submission to the shared task consists of two submissions for Subtask 1 (Inflection): the random string autoencoder for the medium and high data settings of the restricted (main) track and the corpus word autoencoder for the medium data setting of the bonus track (track with external monolingual corpora). All systems use 16K autoencoder examples and an ensemble of three training runs with majority voting.

In the high resource setting of the restricted track, our system outperforms all 17 other submissions, with an average test set accuracy of 95.32% over 52 languages. In the medium resource setting, among 18 submissions our system takes the 6th place with 81.02% (1.78% below the top system). In the medium resource setting of the bonus track, among 2 submissions our system comes first with 82.37%.

2 Baseline MED System

For our baseline system (henceforth **MED-baseline** or **MED**), we use the sequence encoder-decoder architecture and input/output format of the 2016 Shared Task winner (Kann and Schütze, 2016). The architecture follows Bahdanau et al. (2015): that is, the encoder is a bidirectional gated recurrent neural network (GRU) with attention, and the decoder is a uni-directional GRU. For details, see Kann and Schütze (2016) and Bahdanau et al. (2015).

The input sequence consists of space separated characters of the input lemma—the dictionary form of the word—followed by space separated morphological tags, each prepended with a tag marker, for example:

```
w a l k t=V t=V.PTCP t=PRS    (1)
```

The target output is a sequence of characters forming the inflected word, e.g., `w a l k i n g`.

3 Augmenting the Training Data

For the Shared Task, the main competition permits no additional resources beyond the labeled training examples given for each setting. However, Wikipedia dumps in each language are provided for teams who wish to explore semi-supervised methods as well. We examine two methods that use only the training data, and two that also incorporate corpus data. Finally, we explore a method that uses multilingual resources.

3.1 No Outside Resources (AE-TD, AE-RS)

Morphologically related words are typically similar in form, and in many cases, parts of the word are copied from the lemma to the inflected form. As suggested by Kann and Schütze (2017), we hypothesized that training MED to copy strings as a secondary task would help with the morphological inflection task. That is, we train the model simultaneously on the tasks of morphological reinflection and sequence autoencoding, interspersing inflection training examples and autoencoding examples. This can be viewed as a form of multitask learning,[3] and is equivalent to maximizing the log-likelihood for both tasks:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(l,t,w)\in\mathcal{D}} \log p_{\boldsymbol{\theta}}\left(w \mid e(l,t)\right) + \sum_{s\in\mathcal{S}} \log p_{\boldsymbol{\theta}}(s \mid e(s)), \quad (2)$$

where $\mathcal{D}$ is the labeled training data, with each example consisting of a lemma l, a morphological tag t and an inflected form w, and $\mathcal{S}$ is a set of autoencoding examples. The function e represents the encoder, which depends on $\boldsymbol{\theta}$.

In the setting with no outside resources we experiment with two variants of the sequence autoencoder. The first of these, **AE-TD**, uses the

[2] Of course, the morphological variants we find may be noisy, so a better method for identifying these might still improve upon random strings.

[3] Multitask learning for NLP using encoder-decoder networks typically assumes that the separate tasks either have distinct encoders, distinct decoders, or both (e.g., Luong et al., 2016; Alonso and Plank, 2017; Bollmann et al., 2017). Here, we use the same encoder and decoder for both tasks. In preliminary experiments, we tried pre-training the autoencoder instead (see, e.g., Dai and Le, 2015; Kamper et al., 2015), but found that interspersing examples gave a clear advantage.

lemmas and target forms in the training data as inputs to the autoencoder, yielding up to twice as many autoencoder inputs as inflection training pairs (any duplicate lemmas or target forms are included only once).

Our second autoencoder variant, **AE-RS**, uses randomly generated strings as inputs, which means we can produce an arbitrary number of autoencoding examples. In this and following systems, we use the postfix XXK (e.g. 1K, 2K, 4K) to indicate the number of additional examples generated. To obtain each example, we first choose its length uniformly at random from the interval [4, 12] and then sample each character uniformly at random from the alphabet of the respective language.

Input/Output Format We generally follow the input representation outlined in §2, except that morphological tags are replaced with one special tag that stands for autoencoding. The output format does not change.

3.2 Corpus Word Autoencoder (AE-CW)

In the setting where corpus data is available we can replace randomly generated strings with corpus words (system **AE-CW**). We hypothesized that autoencoding words from the actual language would give not only the benefit of learning to copy, but also the benefit of learning the character distributions typical of a given language.

To select words for the corpus word autoencoding task we use Wikipedia text dumps provided for the shared task. We filter out all words shorter than 4 characters. For each language we learn its alphabet from the letters that occur in training words of the original SIGMORPHON training sets and filter out all words that contain foreign characters. We use the remaining words to sample uniformly without repetition the required amount of autoencoding examples. The input and output formats are the same as for the previous approaches.

3.3 Data Mining for Inflected Pairs (DM)

Our next method, **DM**, mines the corpus data to create new training examples by (a) inferring new lemma-inflected form pairs and (b) predicting the tags of the inflected forms. We describe each step below.

Inferring Lemma-Word Form Pairs Although most work on unsupervised learning of morphology has focused on decomposing words into morphemes, their constituent parts, e.g., (Kurimo et al., 2010; Hammarström and Borin, 2011), others have focused on finding morphologically related words and the orthographic patterns relating them (Schone and Jurafsky, 2000; Baroni et al., 2002; Neuvel and Fulop, 2002; Soricut and Och, 2015):

Word 1⇔Word 2	Sim.	Dif.
deceive⇔deception	*dece*	*ive* ⇔ *ption*
receive⇔reception	*rece*	*ive* ⇔ *ption*
perceive⇔perception	*perce*	*ive* ⇔ *ption*
conceive⇔conception	*conce*	*ive* ⇔ *ption*

Table 2: Pairs of related words, their similarities and the respective differences.

$$\text{walk} \Leftrightarrow \text{walking} \quad \varepsilon \Leftrightarrow \text{ing} \tag{3}$$

We adopt the algorithm by Neuvel and Fulop (2002) to learn Word Formation Strategies (WFS)—frequently occurring orthographic patterns that relate whole words to other whole words. The input of this algorithm is a list of N words[4]. The algorithm works by comparing each of the N words to all other words. It first finds word *similarities* as the Longest Common Subsequence (LCS) between the two words. Then it finds word *differences* as the orthographic differences with respect to *similarities* (see Table 2 for examples). Finally, all word pairs with the same *differences* have their *similarities* and *differences* merged into one WFS. For example, words in Table 2 sanction the following WFS:

$$*\#\#\textbf{ceive} \Leftrightarrow *\#\#\textbf{ception} \tag{4}$$

where * and # stand for the optional and mandatory character wild cards respectively. The interpretation of the WFS in Example 4 is: "a word that ends with *ceive* preceded by 2 to 3 characters predicts another word ending with *ception* preceded by the same 2 to 3 characters" (Neuvel and Fulop, 2002). Table 1 gives English WFS examples and sample word pairs that warranted their creation.

Tag Prediction To perform labeling we make use of two resources: (i) a word embedding based part of speech (POS) classifier; (ii) the WFS we learned previously.

[4]We choose N to be 40k and take the 40k most frequent words from each language's Wikipedia dump.

No	WFS	Examples
1	*********####d ⇔ *********####s	invited-invites, wired-wires, tried-tries
2	********####ed ⇔ ********####ing	trailed-trailing, folded-folding, melted-melting
3	********####e ⇔ ********####ing	violate-violating, liberate-liberating, raise-raising
4	**********####s ⇔ **********####al	comics comical, clinics-clinical, corrections-correctional

Table 1: Examples of word formation strategies and sample word pairs that warranted them.

Each of the original shared task training examples contains two word forms and a set of morphological tags including word POS information (see Example 1). We can use words with POS labels and word embeddings to train a POS classifier. Namely we train a support vector machine (SVM) (Cortes and Vapnik, 1995) to predict POS labels using word embeddings as features. We train word embeddings on the Wikipedia text dumps provided for the task. We use FastText[5] by Bojanowski et al. (2016) to train 300 dimensional continuous bag of words embeddings (Mikolov et al., 2013) for all words occurring at least 5 times.

For each training example in the SIGMORPHON training data we examine each of the previously learned WFS. If the training example fits the WFS's orthographic constraints, we examine all word pairs that warranted the creation of this WFS. For a word pair to be labeled with the same morphological tags as the training example we require that both words are classified with the same part of speech as the original training example.

Input/Output Format To encode the additional training examples we use the same format as for the original ones (see §2) except we add a tag which signals that this example has been automatically extracted. We do this as a measure of caution to avoid ambiguities introduced by potentially erroneous training examples.

3.4 Using Multilingual Resources (MLT)

Recently, Kann et al. (2017) showed that training on a high-resource language can improve morphological inflection on a related low-resource language using an encoder-decoder system like the one here. This can be done using the same network as above, but training on inflection examples from multiple different languages—yet another form of multitask learning, since the model parameters are shared between languages. Following Kann et al. (2017), our multilingually trained system (**MLT**) uses the same input format as above, but with an additional tag prepended to each example indicating which language it is from.

Our setup is slightly different from that of Kann et al. (2017), since they had only 50 or 200 examples in each target language, and used only one or two other higher-resource languages for transfer. Here, we have similar amounts of data for each language (1000 examples in most cases), and use larger groups of related languages to train together (see §4).

4 Experiments

Datasets The official shared task data consists of sets for 52 different languages, of which 40 were released as development languages. We used these for our preliminary experiments to compare different systems and quantities of additional training data. The remaining 12 "surprise" languages were released shortly before the test phase of the shared task, and we report results for our best systems on these as well.[6] In the "medium" training data setting, which we focus on in this work, 1000 training instances are given for each language, except for Scottish Gaelic with only 681 instances. Additionally, development sets with 1000 instances are available for all languages except for Basque, Bengali, Haida, Welsh with 100 and Scottish Gaelic with 50.

MED Parameters In our experiments we use the same training method and hyper parameter settings as suggested by Kann and Schütze (2016). Namely, we use 100 hidden units for the encoder and decoder GRUs; 300 dimensions for encoder and decoder embeddings. For training we use stochastic gradient descent, Adadelta (Zeiler, 2012), with gradient clipping threshold of 1.0 and mini batch size of 20. When making predictions

[5] https://github.com/facebookresearch/fastText/

[6] A detailed list of languages can be found at https://sites.google.com/view/conll-sigmorphon2017/task, or see Figure 2.

we use beam-search decoding with a beam of size 12.

Baselines We compare our results with two baselines: the **SIGMORPHON** baseline[7] and **MED** baseline (see §2).

Additional Training Data We train the AE-CW, AE-RS and DM systems with increasing amounts of data: 1k, 2k, or 4k additional training examples. Four thousand training instances is the maximum number of mined training examples available for most languages. We train additional AE-CW and AE-RS systems with 8k and 16k additional training examples to see if any further performance gains are obtainable.

Multilingual Training We train together languages that belong to the same language family. This results in the following groupings: **Baltic** (Latvian, Lithuanian), **Celtic** (Scottish Gaelic, Welsh, Irish), **Germanic** (Danish, Dutch, English, Icelandic, Faroese, German, Swedish, Norwegian-Nynorsk, Norwegian-Bokmal), **Finnic** (Finnish, Estonian), **Iranian** (Persian, Sorani, Kurmanji), **Romance** (Catalan, French, Italian, Portuguese, Spanish, Romanian), **Semitic** (Arabic, Hebrew), **Slavic** (Bulgarian, Czech, Lower-Sorbian, Macedonian, Polish, Russian, Serbo-Croatian, Slovak, Slovene, Ukrainian).

We perform two different multilingual training experiments: First, we train **MLT** without any additional data. In this setting MED trained on individual languages gives a lower bound—MLT should work no worse than MED. We also train **MLT-AE-TD** in which the original training data is used to obtain up to twice as many autoencoding training examples for each language. In this setting the training file of Slavic languages, for example, contains 30K training examples (3K for each language). Here, AE-TD trained on individual languages gives a lower bound on the expected performance. In this setting we also train another stronger and fairer baseline (**AE-TD-RS-XK**), for which training files for individual languages contain exactly the same number of training examples as in the multilingual training case. For example, the training file for MLT-AE-TD when trained on the Baltic branch has 6k training examples—1k original and 2k AE-TD examples for each of the two Baltic languages (Latvian and Lithuanian). So, the Latvian training file for AE-TD-RS-3K also contains 6K training examples: 1K original and 2K training data autoencoding examples, plus an additional 3K random string autoencoding examples, which substitute for the absent Lithuanian training examples. The Lithuanian training file is constructed analogously.

[7] SIGMORPHON baseline: https://github.com/sigmorphon/conll2017/tree/master/evaluation

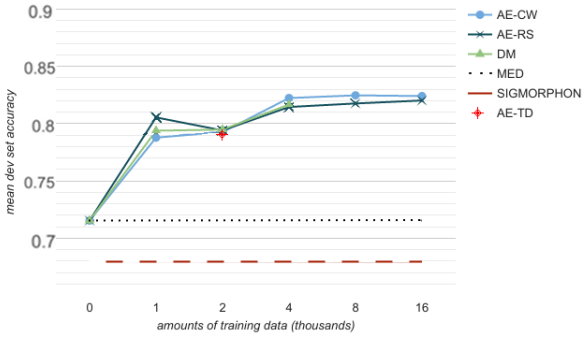

Figure 1: The average performance over the 40 development languages for each of the methods, as a function of the number of added examples.

System	Mean Accuracy
SIGMORPHON	65.67
MED	70.52
AE-TD	77.73
AE-CW-16K	**81.20**
AE-RS-16K	80.40

Table 4: The average performance over all languages except Haida and Khaling (see text).

5 Results and Discussion

5.1 Number of Added Examples

Figure 1 shows how the average performance over the 40 development languages varies with the number of added autoencoder examples, for AE-CW, AE-RS and DM methods. For all three methods, adding more data up to 4k examples helps a lot, after which performance rises more slowly and mostly levels off by about 16k extra examples. (These results contrast with the inflection results from the semi-supervised variational sequence autoencoder of Zhou and Neubig (2017), where even after 150k unlabeled examples, performance still appears to be increasing.) After controlling for the amount of additional data, we see only a small benefit from autoencoding corpus words (AE-CW) rather than random strings (AE-

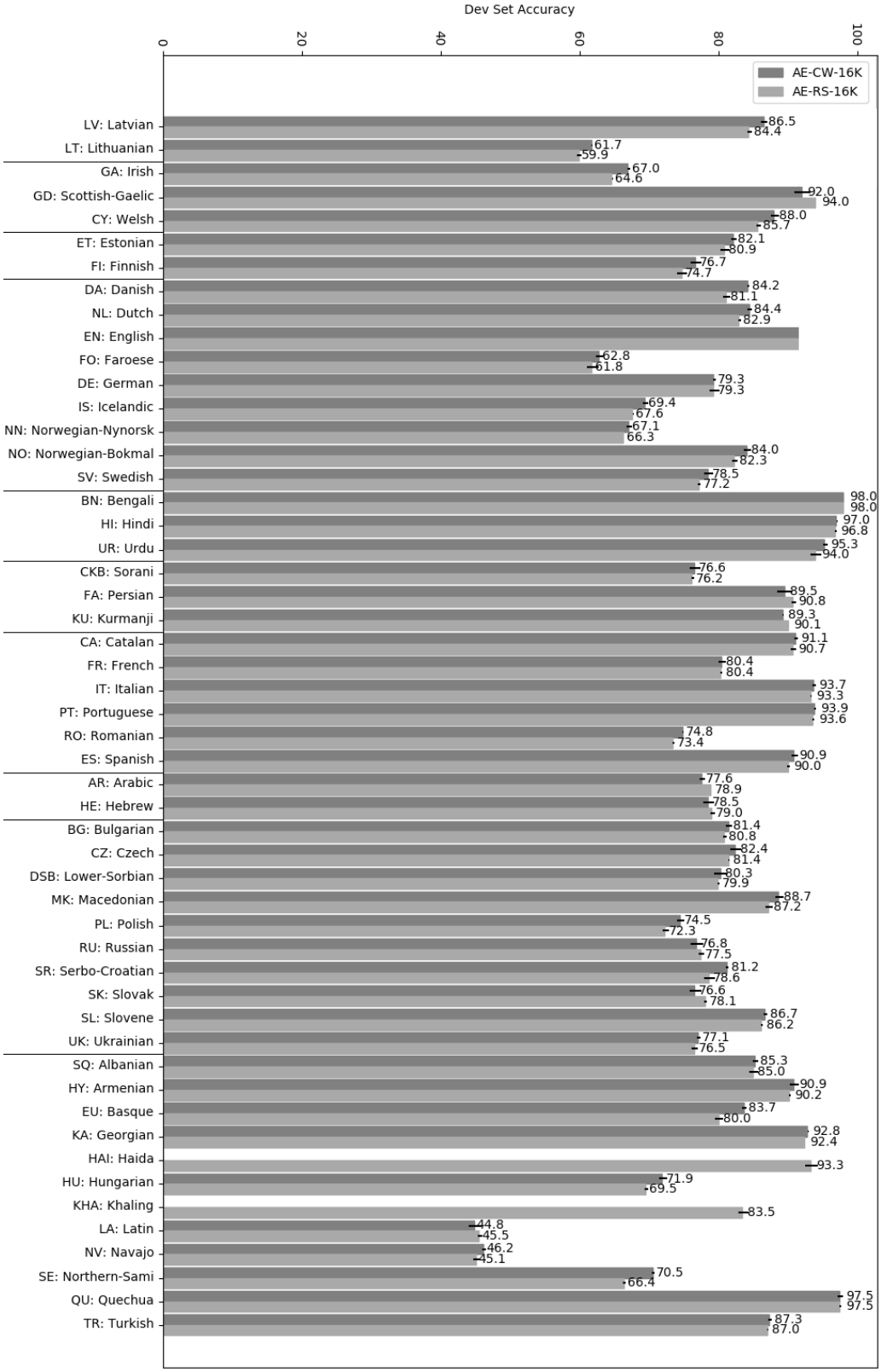

Figure 2: The accuracy of our best systems on all languages. We report the mean and standard error (shown as error bars) across three separate training runs for each language. Languages are grouped by their language families. From the top: Baltic, Celtic, Germanic, Finnic, Iranian, Romance, Semitic, Slavic and miscellaneous languages.

	MED	MLT	AE-TD	MLT-AE-TD	AE-TD-RS-XK
Baltic	63.25	**64.45**	69.15	70.80	**72.20**
Celtic	53.65	**56.09**	63.74	65.22	**68.17**
Germanic	66.53	**73.29**	73.83	75.87	**76.71**
Finnic	59.45	**68.65**	73.35	75.90	**76.77**
Iranian	75.83	**80.83**	81.53	83.60	**84.87**
Romance	80.72	**83.65**	84.90	86.10	**86.67**
Semitic	**71.60**	69.05	76.70	**77.30**	**77.30**
Slavic	63.78	**77.90**	75.65	**80.31**	79.47
Average	66.85	**71.74**	74.86	76.89	**77.77**

Table 3: Multilingual training. We report average development set accuracies, weighted by the number of examples in each development set. The letter X in AE-TD-RS-XK, stands for the number of random string autoencoding examples added (see §4 for more details).

RS).

Figure 1 suggests that on average, adding 1K, 2K or 4k mined training examples (DM curve) gives a similar performance gain to what adding the same amount of autoencoding examples would give (AE-CW and AE-RS).

To investigate the quality of the mined training examples we conduct an experiment where for each morphological tag in SIGMORPHON training data we pick an alternative lemma and target form among the mined examples with the same tag. We hypothesize that, if correct, mined examples should serve the same function as the original ones and the average performance should not change. On average this resulted in 500 swaps per language. The average MED performance on the modified training sets is about 10% lower than on the original training files.

This suggests, that—although noisy—the mined examples, when annotated with an additional "mining"-tag, work as model regularizers, thus benefiting MED's performance on the inflection task.

Obtaining autoencoding examples, however, is computationally simpler than mining additional training examples, hence given the similar effects the autoencoding approach seems preferable over the data mining.

5.2 Best Monolingual Systems

The average development set accuracy over 50 languages[8] of our best system AE-CW-16K is 81.2% (see Table 4) closely followed by AE-RS-16K with 80.4%. For comparison, this is about 15.5% and 10.6% absolute gain over the SIGMORPHON and MED baselines respectively. AE-TD on average performs only 3.5% worse than AE-CW-16K although using 87.5% fewer autoencoding examples and no external resources. Figure 2 shows the accuracy of our best systems on all languages. We report the average accuracy and standard error across three separate training runs for each language.

We conducted a paired-samples t-test to compare mean development set accuracies for 50 languages between AC-CW-16K and AC-RS-16K systems, using 3 runs each. The test suggested that there was a significant difference between accuracies of AC-CW-16K and AC-RS-16K ($T(49) = 4.04$, $p < 0.01$), although the difference is small relative to the gains over the baselines.

5.3 Multilingual Training

Table 3 shows results for multilingual training experiments. Due to the different development set sizes (see §4) we report **weighted average** development set accuracies. MLT without any additional data is better than the MED baseline for most related languages except Semitic languages, on average giving about 7% improvement over the MED baseline. MLT-AE-TD, the system in which the original training data is used to obtain autoencoding training examples on average outperforms the conservative AE-TD baseline by 2.5% absolute, but barely reaches the performance of AE-TD-RS-XK baseline. AE-TD-RS-XK baseline (for details see §4) gives performance for each individual language if all training examples of other languages in an MLT-AE-TD training file are replaced by random string autoencoding exam-

[8] Haida and Khaling do not have Wikipedia text dumps to train **AE-CW** and are thus excluded from all averages.

ples. Table 3 shows that on average MLT-AE-TD works no better than AE-TD-RS-XK. Currently, it is unclear whether the performance gains in MLT and MLT-AE-TD experiments are due to knowledge transfer from related languages as suggested by Kann et al. (2017), or because different languages serve as model regularizers with respect to each other. Performance on 6 out of 8 groups of related languages, however, suggests that random string autoencoding is not only simpler but also a better performing method than multilingual training.

5.4 Shared Task Submission: Test Results

We submitted two final systems to the medium-resource track of Subtask 1 (Inflection): **AE-RS-16K** for the restricted (main) track, and **AE-CW-16K** for the bonus track (where external monolingual corpora are permitted). The final systems are ensembles of 3 separate training runs, and the final answer is selected by majority voting (or chosen at random in case of a tie).

Although we did not tune any system to the high-resource track, we also submitted results there, using an analogous ensemble system with 16k autoencoder at random strings in addition to the 10k training examples. We did not submit to the low-resource track because the results from the AE-RS-16k method were below the baseline system on the development set.

In the high resource setting of the restricted track, our system achieved an average test set accuracy of 95.32%, which is an 17.51% absolute improvement over the Shared Task baseline, and the top performance of 18 submissions. In the medium resource setting AE-RS-16K gives 81.02%—an improvement of 16.32% absolute over the Shared Task baseline. This result, however, is 1.78% absolute lower than the best system's performance, so among 18 submissions AE-RS-16K takes the 6th place. In the medium resource setting of the bonus track among 2 submissions AE-CW-16K comes first with 82.37%. This is 1.35% better than our restricted track submission (AE-RS-16K), but still 0.43% worse than the top performing system in the restricted track.

6 Conclusion

We evaluated several ways to improve the morphological inflection performance of a state-of-the-art encoder-decoder model (MED) when relatively few labeled examples are available. In experiments on 52 languages, we showed that all methods considerably outperformed the MED baseline. Autoencoding corpus words (AE-CW) gave the largest improvement, but was only slightly better than autoencoding random strings (AE-RS). We found no benefit from cross-lingual knowledge transfer as compared to using an equivalent number of random string autoencoder examples.

Our results suggest that the main benefit of the various data augmentation methods is providing a strong bias towards learning the identity transformation and/or regularizing the model, with a slight additional benefit obtained by learning the typical character sequences in the language. These benefits can be achieved with very simple methods and few or no additional data resources.

References

Héctor Martínez Alonso and Barbara Plank. 2017. Multitask learning for semantic sequence prediction under varying data conditions. In *Proceedings of EACL*. Association for Computational Linguistics.

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

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

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

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning* 20(3):273–297.

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*. Association for Computational Linguistics.

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 ACL*. Association for Computational Linguistics.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*. pages 3079–3087.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics* 37(2):309–350.

Herman Kamper, Micha Elsner, Aren Jansen, and Sharon Goldwater. 2015. Unsupervised neural network based feature extraction using weak top-down constraints. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pages 5818–5822.

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

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of ACL*. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2017. Unlabeled data for morphological generation with character-based sequence-to-sequence models. *arXiv preprint arXiv:1705.06106* .

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010. Morpho challenge competition 2005–2010: Evaluations and results. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*. Association for Computational Linguistics, SIGMORPHON '10, pages 87–95.

Minh-Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of ICLR*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Sylvain Neuvel and Sean A Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*. Association for Computational Linguistics, pages 31–40.

Patrick Schone and Daniel Jurafsky. 2000. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics, pages 67–72.

Radu Soricut and Franz Josef Och. 2015. Unsupervised morphology induction using word embeddings. In *Proceedings of HLT-NAACL*. Association for Computational Linguistics, pages 1627–1637.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

Chunting Zhou and Graham Neubig. 2017. Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction. *arxiv preprint arXiv:1704.01691* .