Data Augmentation for Morphological Reinflection

Miikka Silfverberg, Adam Wiemerslage, Ling Liu and Lingshuang Jack Mao
Department of Linguistics
University of Colorado Boulder
first.last@colorado.edu

Abstract

This paper presents the submission of the Linguistics Department of the University of Colorado at Boulder for the 2017 CoNLL-SIGMORPHON Shared Task on Universal Morphological Reinflection. The system is implemented as an RNN Encoder-Decoder. It is specifically geared toward a low-resource setting. To this end, it employs data augmentation for counteracting overfitting and a copy symbol for processing characters unseen in the training data. The system is an ensemble of ten models combined using a weighted voting scheme. It delivers substantial improvement in accuracy compared to a non-neural baseline system in presence of varying amounts of training data.

1 Introduction

Natural language processing (NLP) for English is typically word based, that is, words such as **dogs**, **cat's** and **they've** are treated as atomic units. In the case of English, this is a viable approach because lexemes correspond to a handful of inflected forms. However, for languages with more extensive inflectional morphology, the approach fails because one lexeme can be realized by thousands of distinct word forms in the worst case. Therefore, NLP systems for languages with extensive inflectional morphology often need to be able to generate new inflected word forms based on known word forms. This is the task of *morphological reinflection*.

The traditional approach to word form generation is rule-based. For example, finite-state technology has been successfully applied in constructing morphological analyzers and generators for a large variety of languages (Karttunen and Beesley, 2005). Unfortunately, the rule-based approach is labor-intensive and therefore costly. Additionally, coverage can become a problem because systems need to be continually updated with new lexemes. For these reasons, machine learning approaches have recently gained ground.

Results from the 2016 SIGMORPHON Shared Task on Morphological Reinflection (Cotterell et al., 2016) indicate that models based on recurrent neural networks can deliver high accuracies for reinflection. The winning system by Kann and Schütze (2016) achieved an average accuracy in excess of 95% when tested on 10 languages.[1] Based on these results, morphological reinflection could be considered a solved problem. However, the 2016 shared task employed training sets of more than 10,000 word forms for most languages. In a setting with less training data, the reinflection task becomes much more challenging. In an extreme low-resource setting of 100 training examples, a standard RNN Encoder-Decoder system like the one used by Kann and Schütze (2016) will typically perform quite poorly.[2]

This paper documents the submission of the CU Boulder Linguistics Department for the 2017 CoNLL-SIGMORPHON Shared Task on Universal Morphological Reinflection. The task covers 52 languages from different language families with a wide geographical distribution. The task evaluates systems trained on varying amounts of data ranging from 100 to more than 10,000 training examples.

Our system is an RNN Encoder-Decoder (Cho et al., 2014) specifically geared toward a low-resource setting. The system closely resembles the

[1]For inflecting lemmas according to a given morphological feature set.

[2]According to experiments performed by the authors, the system employed by Kann and Schütze (2016) delivered accuracies between 0% and 1% for most languages in the shared tasks when using 100 training examples.

system introduced by Kann and Schütze (2016). However, the novelty of our approach lies in the training procedure. We augment the training data with generated training examples. This is a commonly used technique in image processing but it has been employed to a lesser degree in NLP. Data augmentation counteracts overfitting and allows us to learn reinflection systems using small training sets.

We employ an ensemble of 10 models under a weighted voting scheme. We also implement a mechanism, the copy symbol, which allows the system to copy unseen characters from an input lemma to the resulting word form. This improves accuracy for small training sets. Unfortunately, due to time constraints, we were only able to use the copy symbol in Task 2 of the shared task.

For Task 1 of the shared task, we achieve substantial improvements over a non-neural baseline (Cotterell et al., 2017), even in the low resource setting.

The paper is organized as follows: Section 2 presents related work on morphological reinflection and data augmentation for natural language processing. In Section 3, we describe the shared task and associated data sets. We provide a detailed description of our system in Section 4 and present experiments and results in Section 5. Finally, we provide a discussion of results and conclusions in Section 6.

2 Related Work

Several existing approaches to morphological reinflection are based on traditional structured prediction models. For example, Liu and Mao (2016) and King (2016) use Conditional Random Fields (CRF) and Alegria and Etxeberria (2016) and Nicolai et al. (2016) employ different phoneme-to-grapheme translation systems. Other approaches include learning a morphological analyzer from training data and applying it to reinflect test examples (Taji et al., 2016) and extracting morphological paradigms from the training data which are then applied on test words (Ahlberg et al., 2015; Sorokin, 2016). The results of the 2016 SIGMORPHON Shared Task on Morphological Reinflection indicate that none of these approaches can compete with deep learning models. The deep learning systems outperformed all other systems by a wide margin.

The three best performing teams (Kann and Schütze, 2016; Aharoni et al., 2016; Östling, 2016) in the 2016 SIGMORPHON shared task employed deep learning approaches based on the RNN Encoder-Decoder framework proposed by Cho et al. (2014) and later used for machine translation by Bahdanau et al. (2014). This family of models is intuitively appealing for morphological reinflection because of the obvious parallels between the reinflection and translation tasks. The success of the winning system by Kann and Schütze (2016) highlights the importance of an additional attention mechanism introduced by Bahdanau et al. (2014).

Although the RNN Encoder-Decoder framework has proven to be highly successful in morphological reinflection, an out-of-the-box RNN Encoder-Decoder system performs poorly in presence of small training sets due to overfitting. To alleviate this problem, we employ data augmentation, that is, augmentation of the training set with artificial, generated, training examples. The technique is well known in the field of image processing (Krizhevsky et al., 2012; Chatfield et al., 2014). Even though the technique is used less frequently in NLP, a number of notable approaches do exist. Sennrich et al. (2016) use monolingual target language data to improve the performance of an Encoder-Decoder translation system. They first train a translation system from the target language to the source language, which is used to back-translate target language sentences to source language sentences. The sentence pairs consisting of a translated source sentence and a genuine target sentence are then added to the training data. Other approaches to data augmentation in NLP include substitution of words by synonyms (Fadaee et al., 2017; Zhang and LeCun, 2015) and paraphrasing.

3 Task Description and Data

The shared task consists of two subtasks: (1) generation of word-forms based on a lemma and a set of morphological features (for example, **dog+N+Pl** $\rightarrow$ **dogs**), and (2) completion of morphological paradigms given a small number of known forms (see Figure 1).

Systems are evaluated on 52 languages.[3]

[3]Albanian, Arabic, Armenian, Basque, Bengali, Bokmal, Bulgarian, Catalan, Czech, Danish, Dutch, English, Estonian, Faroese, Finnish, French, Georgian, German, Haida, Hebrew, Hindi, Hungarian, Icelandic, Irish, Italian, Khaling, Kurmanji, Latin, Latvian, Lithuanian, Lower Sorbian, Mace-

For both subtasks and all languages, there are three data settings, which differ with respect to the size of available training data: *low*, *medium*, and *high*. In Task 1, these span 100, 1,000 and 10,000 examples respectively. However, there is no training set for the high setting for Gaelic. In Task 2, there are 10 example paradigms in the low setting. Most languages have 50 example paradigms in the medium setting (Basque has 16, Haida 21 and Gaelic 23). In the high settings, most languages have 200 example paradigms (Bengali has 86, Urdu 123 and Welsh 133). There is no training set for the high setting in Task 2 for Basque, Haida and Gaelic. All settings use the same development and test sets. Further details concerning the shared task and languages can be found in Cotterell et al. (2017).

lock	–	V V.PTCP PRS
lock	–	V 3 SG PRS
lock	–	V V.PTCP PST
lock	lock	V NFIN
lock	–	V PST

Figure 1: Illustration of Task 2 – the paradigm completion task. The system will fill in missing forms based on the lemma, morphological features and the known word forms.

4 System Description

Our system is an RNN Encoder-Decoder network heavily influenced by Kann and Schütze (2016). The key difference is that our system is trained using augmented data, which substantially improves accuracy given small training sets. We train several models and employ a weighted voting scheme, which improves results upon a baseline majority voting system. Additionally, we use copy symbols which allow the system to process lemmas that contain characters that were missing in the training data.

4.1 RNN Encoder-Decoder with Attention

We use an RNN Encoder-Decoder model with attention proposed by Bahdanau et al. (2014) for machine translation, which was later applied to morphological reinflection by Kann and Schütze (2016). The architecture of our model differs from the model proposed by Kann and Schütze (2016) only with regard to minor details.

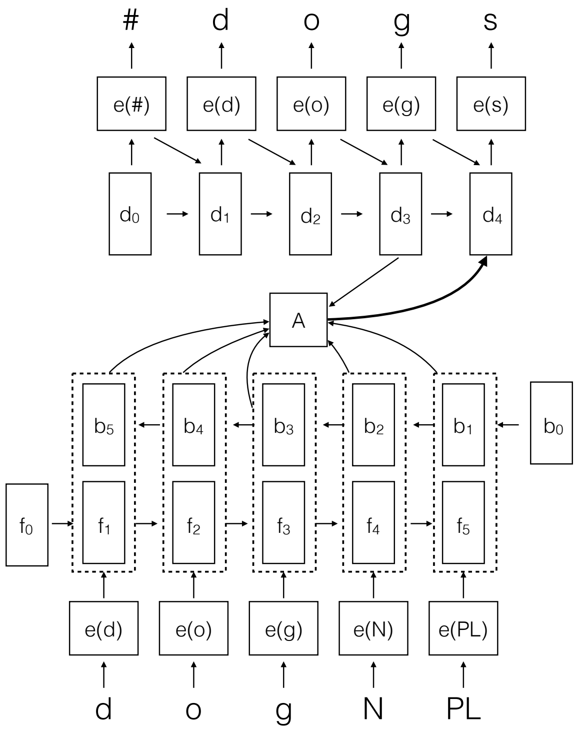

Figure 2: The RNN Encoder-Decoder for morphological reinflection. The system takes a lemma and associated tags as input and produces an output form.

The high-level intuition of the system is conveyed by Figure 2. The system takes a sequence of lemma characters and morphological features as input (for examples **d**, **o**, **g**, **N**, **PL**) and produces a sequence of word form characters as output (**d**, **o**, **g**, **s**). It incorporates two encoder LSTMs, which operate on embeddings of input characters and morphological features. One of the encoders consumes the input lemma and features from left to right and the other one consumes them from right to left. This results in two sequences of state vectors, which are translated into a sequence of output characters by a decoder LSTM with an attention mechanism.

More specifically, our system first computes character embeddings $\mathrm{e}(\cdot)$ for input characters and features. These embeddings are then encoded into forward state vectors f_i and backward state vectors b_i by a bidirectional LSTM (a combination of a forward and backward LSTM). Each forward and backward state pair $e_i = (\mathrm{f}_i, \mathrm{b}_i)$ is used as the bidirectional LSTM state at position i. Subsequently, a decoder LSTM generates a sequence of embeddings which is then transformed into output characters by a softmax layer. At each state

donian, Navajo, Northern Sami, Nynorsk, Persian, Polish, Portuguese, Quechua, Romanian, Russian, Gaelic, Serbo-Croatian, Slovak, Slovene, Sorani, Spanish, Swedish, Turkish, Ukrainian, Urdu, and Welsh.

during decoding, the current state vector of the decoder is computed based on (1) the previous decoder state, (2) the previous output embedding, and (3) all encoder states $(\mathrm{f}_i, \mathrm{b}_i)$. The simultaneous use of all encoder states is realized by an attention mechanism A which computes a weight $\mathrm{w}_{i,j-1}$ for each encoder state e_i given the previous decoder state f_{j-1}. These weights are then normalized into weighting factors $\epsilon_{i,j-1}$ using softmax, that is $\epsilon_{i,j-1} = \exp(\mathrm{w}_{i,j-1})/\sum_{i=1}^{n}\exp(\mathrm{w}_{i,j-1})$. The next decoder state f_j is then determined based on the previous decoder state f_{j-1}, the previous output embedding and a weighted average of all encoder states $\mathrm{A}(\mathrm{f}_{j-1}, e_1, ..., e_n)$ given in Equation 1.

$$\mathrm{A}(\mathrm{f}_{j-1}, e_1, ..., e_n) = \sum_{i=0}^{n} \epsilon_{i,j-1}\mathrm{e}_i \qquad (1)$$

The attention mechanism A is implemented as a feed-forward neural network with one hidden layer and hyperbolic tangent non-linearity (tanh).

For the encoders and the decoder, we use 2-layer LSTMs (Hermans and Schrauwen, 2013) with peephole connections (Gers and Schmidhuber, 2000) and coupled input and forget gates (Greff et al., 2015). We train our system using Stochastic Gradient Descent. Our system is implemented using the Dynet toolkit (Neubig et al., 2017)[4] and our code is freely available.[5]

There are three hyper-parameters in our system: the character embedding dimension, the size of the hidden layer of the LSTM models and the size of the hidden layer of the attention network. We set these to 32 for most languages but use 100 for a number of languages, as explained in Section 5.

4.2 Data Augmentation

In order to counteract overfitting caused by data sparsity in the low and medium data settings of the shared task, we use data augmentation. That is, we generate new training examples from existing training examples.

Our data augmentation technique is based on the observation that in most cases word forms can be split into three parts: an inflectional prefix, a word stem and an inflectional suffix. For example, the English word **fizzling** can be split into **0+fizzl+ing**. In many cases, as in the case of the lemma **fizzle** and word form **fizzling**, the stem is shared between the lemma and word form. By replacing it, in both the lemma and word form, with another string, we can produce a new training example from an existing one. For instance, we can produce a new example (**sfkekgivlofe+V+PRS+PCP**, **sfkekgivlofing**) from (**fizzle+V+PRS+PCP**, **fizzling**) by replacing **fizzl** with **sfkekgivlof**.

Data augmentation requires that we can identify word stems. We approximate this by identifying the longest common continuous substring of the word form and lemma. This strategy can be expected to work well for languages with largely concatenative morphology. In languages with extensive stem changes or stem allomorphy, it can, however, fail.

We experimented with two different techniques for generating new stems:

- Draw each character from a uniform distribution over the set of characters occurring in the training file.
- First, train a language model on the training data. Then, use a sampling-based method to identify a likely character sequence c_1, ..., c_m based on the probability given by the language model to the string $p_1...p_l c_1...c_m s_1...s_n$, where $p_1...p_l$ and $s_1...s_l$ are the inflectional prefix and suffix respectively.

We experimented with two different language models—a simple trigram based model with additive smoothing and a 5-gram model with Witten-Bell smoothing (Witten and Bell, 1991).

The augmented training data generated using the language models seems to be phonotactically superior to the data generated by the uniform distribution over all characters. However, surprisingly, it fails to produce comparable accuracy. Therefore, we only report results for the strings drawn from the uniform distribution.

4.3 Voting

For each language and setting, we train an ensemble of ten models. The most straightforward way of utilizing such an ensemble is *majority voting* which is employed by Kann and Schütze (2016). In majority voting, the output candidate which was generated by the greatest number of models is the final output of the ensemble. In contrast to Kann

[4] http://dynet.readthedocs.io/en/latest/index.html
[5] https://github.com/mpsilfve/conll2017

and Schütze (2016), we apply a *weighted voting scheme* to the model ensemble.

In weighted voting, each model receives a weight $w_i \in [0, 1]$. It then uses this weight to vote for the output candidate that it generated. Let S_j be the set of models that generated output candidate c_j. Then the total weight W_j of candidate c_j is given by Equation 2. The candidate with the highest total weight is the output of the ensemble. It is easy to see that setting all model weights $w_i = 1/10$ gives regular majority voting.

$$W_j = \sum_{i \in S_j} w_i \tag{2}$$

We tune model weights using Gibbs sampling in order to attain improved accuracy. Gibbs sampling is implemented as a function which iteratively adjusts the weight distribution $\{w_1 ... w_{10}\}$ in order to find weights that result in improved accuracy on the development set. Each adjustment is made by moving some probably mass of size α from a randomly selected weight w_i onto another randomly selected weight w_j as illustrated in Figure 3.[6] The new weight distribution is then accepted or rejected based on the resulting development set accuracy. We initialize the weights using an even distribution, where $w_i = 1/10$.

The development set accuracy a_2 of the adjusted weight distribution is checked against the development set accuracy a_1 of the previous distribution, and the adjusted distribution is accepted with a probability proportional to a_2/a_1. This draws upon the intuition of Gibbs Sampling that an inferior configuration is sometimes accepted in order to account for the non-convex nature of the objective function.

After Gibbs sampling completes, the weight distribution attaining maximal development set accuracy $w_{max} = \{w_1 ... w_{10}\}$ is returned.

4.4 The Copy Symbol

The decoder of an RNN Encoder-Decoder system can only emit characters that were observed in the training data. This is typically a minor problem when using large training sets because these are likely to contain all frequent orthographic symbols. However, it can become a severe problem when the training set is very small. The problem

[6]We test α values in the set {.001, .01, .05, .1, .2} and run Gibbs sampling for 10,000 iterations.

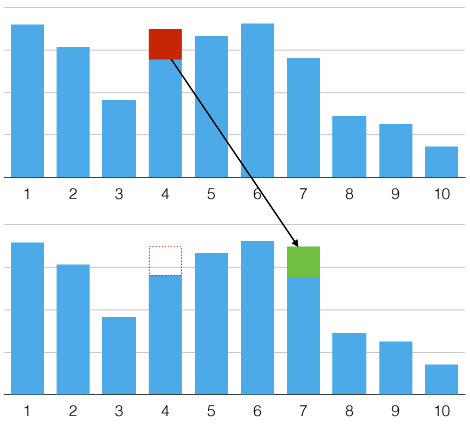

Figure 3: The probability mass is moved from model 4 to model 7 in order to test a new weight distribution

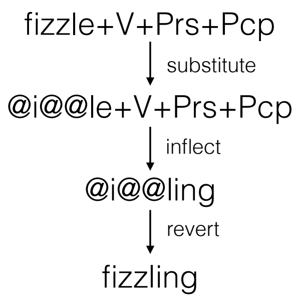

Figure 4: Substitution of unknown characters with copy symbols (@), inflection, and subsequent reversion. In this example, the characters **f** and **z** are missing from the training data.

can have a surprisingly large effect on overall accuracy because reinflection will often fail when even one of the characters in the lemma is unknown to the system.

In order to solve the problem of missing characters, we use a special *copy symbol*. During test time, unknown symbols are substituted by copy symbols and reinflection is performed. After reinflection, each copy symbol is reverted back to the original unknown symbol as shown in Figure 4. Reversion is performed by substituting the ith copy symbol in the output string with the ith unknown symbol in the lemma. If extra copy symbols remain after reversion, they are replaced with the empty string.

Generated stems with copy symbols are added to the training data during data augmentation. This allows the system to learn to copy the symbols from the input lemma to the output word form.

5 Experiments and Results

For Task 1, we train ten models for each language and setting. We then apply weighted voting as explained in Section 4. For most languages, a hidden layer size, embeddings size, and attention layer size of 32 gave reasonable results. For 11 languages, **Faroese, French, German, Haida, Hungarian, Icelandic, Latin, Lithuanian, Navajo, Bokmal**, and **Nynorsk**, we found 32 insufficient, and set hidden layer size, embedding size and attention layer size to 100 instead. Setting the layer size to 100 might improve results for other languages as well. Unfortunately, we did not have enough time to test this.

Data augmentation is used in order to improve accuracy in the low and medium training data settings for Task 1. In the low setting, we add 4900 augmented training examples to the training set, and in the medium data setting, we add 9900 augmented training examples. Given that the original low training data spans 100 and the medium training data spans 1000 examples, this means that the original training data accounts for 2% of the augmented low training set and 10% of the augmented medium training set.

For Task 2, we also use augmented data. In the low setting, we add augmented data until the total size of the training set is 20,000 examples. In the medium and high settings, we add augmented examples until the size of the training set is 25,000. Time constraints prohibited us from using more generated data.

Because of the large variance of the sizes of training sets in Task 2 (for example the low Basque training data spans 4,750 examples, whereas the low English training data spans 50 examples), some languages use substantially more augmented data than other languages. In the high setting, some languages, in fact, draw upon no augmented data at all due to the large size of the training set.

For Task 2, we use the copy symbol as explained in Section 4. This would probably have resulted in improved accuracy for Task 1 as well. Unfortunately, we were unable to run experiments using the copy symbol for Task 1 because of time constraints.

The test results for Task 1 and Task 2 are shown in Table 1. For Task 1, the RNN system achieves average accuracy 45.74% for the low settings, 77.60% for the medium setting and 92.97% for the high setting. All of these figures are substantially greater than the baseline accuracies which are 37.90%, 64.70% and 77.81% for the different settings, respectively.

The RNN system fails to achieve the baseline accuracy for eight languages in the low settings: Dutch (51.90% versus 53.60%), Haida (24.00% versus 32.00%), Hungarian (16.00% versus 21.00%), Kurmanji (79.50% versus 82.80%), Latvian (62.60% versus 64.20%), Lithuanian (19.80% versus 23.30%), Navajo (11.70% versus 19.00%) and Romanian (43.10% versus 44.80%). Additionally, there is one language in the medium setting where the RNN does not achieve the baseline, namely Danish (76.70% versus 78.10%) and another one in the high setting, namely Quechua (90.30 versus 95.40).

For Task2, the RNN system fails to achieve the baseline accuracy for most languages and settings.

6 Discussion and Conclusions

The experiments clearly demonstrate that the system presented in this paper delivers substantial improvements in accuracy over a non-neural baseline for most of the 52 languages in the shared task and in all data settings in Task 1. Due to data augmentation, it improves upon the baseline even in the extreme low resource setting of a mere 100 training examples. In this setting, a conventional RNN system will overfit the training data and, consequently, generalize poorly. Indeed, we found it impossible to train models for the low training data setting without using data augmentation (all models delivered accuracies in the range 0-1%). In Task 1, we did not apply copy symbols due to time constraints. We estimate that this reduces accuracy for the low setting by about 2%.

Even though our system achieves substantial improvements over baseline in Task 1, there are several languages which do not reach the performance of the baseline system in Task 2. One possible cause for this is overfitting due to insufficient variation in the training set. A single lemma occurs multiple times in the Task 2 training data sets because training examples form complete paradigms, which contain dozens (or even hundreds) of word forms. Additionally, the number of unique lemmas in Task 2 training sets is substantially lower than the number of unique lemmas in Task 1 training sets of the same setting. For example, the low setting Task 1 training data for Finnish contains 100 unique lemmas, whereas the Task 2 data set only contains 10 unique lemmas. Finally, time constraints prevented us from training a model ensemble for Task 2. This would probably have improved accuracy for several lan-

	Task 1						Task 2					
	Low		Medium		High		Low		Medium		High	
	RNN	Baseline	RNN	Baseline	RNN	Baseline	RNN	Baseline	RNN	Baseline	RNN	Baseline
Albanian	**31.00**	21.10	**89.40**	66.30	**97.60**	78.90	12.19	12.69	82.17	83.87	86.36	89.46
Arabic	**29.50**	21.80	**73.60**	42.10	**90.40**	50.70	**48.78**	42.85	**63.01**	54.34	**75.54**	55.67
Armenian	**51.30**	35.80	**87.50**	72.70	**96.30**	87.20	75.47	76.18	**86.57**	80.89	**92.04**	86.11
Basque	**4.00**	2.00	**66.00**	2.00	**100.00**	5.00	**1.54**	0.46	**7.35**	4.40	-	-
Bengali	**60.00**	50.00	**95.00**	76.00	**99.00**	81.00	73.38	77.20	19.75	85.86	21.02	87.52
Bulgarian	**57.10**	30.20	**79.90**	72.80	**97.40**	88.80	**35.51**	33.50	49.25	49.58	**78.39**	74.37
Catalan	**66.40**	55.90	**89.50**	84.30	**97.60**	95.50	90.06	94.16	79.69	95.33	90.06	96.03
Czech	**41.90**	39.30	**86.30**	81.50	**92.40**	89.60	16.39	26.56	46.68	56.12	68.36	85.79
Danish	**68.90**	58.40	76.70	78.10	**90.50**	87.80	**53.11**	41.31	64.92	71.15	71.48	75.41
Dutch	51.90	53.60	**74.70**	73.20	**95.60**	87.00	45.57	50.18	60.33	67.71	73.06	78.04
English	**87.80**	80.60	**91.60**	90.90	**95.60**	94.70	**84.40**	76.40	81.60	84.00	84.00	91.60
Estonian	**32.20**	21.50	**74.00**	62.90	**97.10**	78.00	**50.17**	39.81	**61.76**	60.71	**78.29**	77.07
Faroese	**41.20**	30.00	**64.60**	60.60	**85.40**	74.10	46.79	49.78	53.21	59.19	65.62	70.10
Finnish	**15.80**	15.40	**67.20**	43.70	**93.80**	78.20	54.58	60.82	57.14	63.30	**68.78**	68.18
French	**63.00**	61.80	**77.80**	72.50	**88.20**	81.50	84.10	87.09	**85.67**	85.16	89.48	92.63
Georgian	**81.80**	70.50	**92.50**	92.00	**95.40**	93.80	78.86	78.86	81.00	82.42	89.31	90.38
German	**56.60**	54.30	**74.60**	72.10	**89.70**	82.40	68.28	69.83	68.47	70.41	75.82	76.40
Haida	24.00	32.00	**68.00**	56.00	**80.00**	67.00	45.85	47.15	59.63	64.53	-	-
Hebrew	**35.40**	24.70	**77.80**	37.50	**98.50**	54.00	28.83	33.27	**54.89**	42.70	**70.46**	54.09
Hindi	**65.30**	29.10	**93.30**	85.90	**100.00**	93.50	63.62	64.49	61.79	71.11	9.15	96.82
Hungarian	16.00	21.00	**56.20**	42.30	**86.40**	68.50	11.56	17.91	39.68	45.73	**54.95**	53.97
Icelandic	**40.80**	30.30	**67.10**	60.40	**89.10**	76.30	**51.40**	45.79	**56.57**	54.51	63.22	67.36
Irish	**31.30**	30.30	**57.00**	44.00	**88.70**	53.00	26.46	35.95	**44.34**	40.33	**53.28**	47.99
Italian	**56.40**	41.10	**85.90**	71.60	**97.00**	76.90	58.29	66.95	**77.62**	71.86	**89.86**	73.05
Khaling	**10.20**	3.10	**82.90**	17.90	**98.90**	53.70	39.16	42.30	7.53	58.20	**89.64**	79.08
Kurmanji	79.50	82.80	**91.10**	89.10	**94.40**	93.00	65.04	78.43	87.48	88.35	**93.74**	93.39
Latin	**19.30**	16.00	**46.10**	37.60	**80.50**	47.60	22.55	24.45	38.07	39.53	**50.51**	47.58
Latvian	62.60	64.20	**86.00**	85.70	**94.60**	92.10	**74.78**	68.88	**81.99**	79.97	**88.47**	86.46
Lithuanian	19.80	23.30	**58.40**	52.20	**92.90**	64.20	28.19	38.27	61.73	65.92	**64.14**	60.57
Lower Sorbian	**52.30**	33.80	**83.60**	70.80	**95.40**	86.40	27.22	38.20	**71.16**	65.92	80.15	82.27
Macedonian	**59.60**	52.10	**90.40**	83.60	**95.20**	92.10	14.74	42.49	83.12	86.41	**92.56**	89.70
Navajo	11.70	19.00	**40.50**	33.50	**83.10**	37.80	**19.73**	0.00	**32.60**	0.00	**46.30**	0.00
Northern Sami	**18.70**	16.20	**57.00**	37.00	**96.10**	64.00	15.32	15.62	27.93	31.43	**54.61**	45.68
Norwegian Bokmal	**73.80**	67.80	**80.80**	80.70	**91.50**	91.00	**49.06**	41.51	**57.23**	50.94	**70.44**	67.92
Norwegian Nynorsk	**50.50**	49.60	**62.50**	61.10	**87.50**	76.90	39.88	42.33	56.44	60.74	60.74	64.42
Persian	**38.30**	24.50	**86.10**	62.30	**99.50**	79.00	**84.69**	73.42	**94.47**	78.29	25.09	76.44
Polish	**43.70**	41.30	**78.00**	74.00	**90.90**	88.00	55.19	56.72	79.77	80.28	83.10	90.27
Portuguese	**68.40**	63.60	**94.70**	93.40	**99.30**	98.10	89.94	91.71	92.10	95.29	36.23	96.19
Quechua	**30.60**	16.40	**88.20**	70.30	90.30	95.40	79.84	91.33	0.04	91.34	64.45	89.13
Romanian	43.10	44.80	**77.40**	69.40	**85.50**	79.80	10.36	14.20	60.80	61.54	75.00	78.99
Russian	**45.90**	45.60	**81.90**	75.90	**90.80**	85.70	36.66	40.18	82.21	82.98	**87.42**	85.58
Scottish Gaelic	**56.00**	44.00	**74.00**	48.00	-	-	29.96	44.13	**44.53**	41.30	-	-
Serbo-Croatian	**39.20**	18.40	**83.30**	64.50	**92.10**	84.60	27.90	30.07	36.59	36.84	74.40	77.66
Slovak	**46.70**	42.40	**78.00**	72.30	**89.30**	83.30	38.86	44.39	59.00	59.89	68.27	69.16
Slovene	**60.20**	49.00	**86.30**	82.20	**95.80**	88.90	52.15	57.74	**69.15**	67.87	**77.18**	76.48
Sorani	**27.10**	19.30	**71.50**	51.70	**89.10**	63.60	43.53	54.78	67.96	68.30	8.88	72.27
Spanish	**63.60**	57.10	**89.50**	84.70	**96.80**	90.70	79.58	79.75	86.53	92.18	34.63	93.58
Swedish	**60.40**	54.20	**76.30**	75.70	**87.60**	85.40	31.47	43.53	**59.41**	57.35	70.29	78.24
Turkish	**19.70**	14.10	**66.60**	32.90	**96.40**	72.60	**34.89**	20.93	**76.05**	73.26	31.08	85.05
Ukrainian	**50.40**	43.90	**79.70**	72.80	**90.20**	85.40	32.38	43.97	65.71	67.14	72.54	73.97
Urdu	**64.60**	31.70	**96.10**	87.50	**98.30**	96.50	79.56	80.59	67.23	81.02	90.79	95.33
Welsh	**53.00**	22.00	**82.00**	56.00	**98.00**	69.00	**82.72**	51.67	79.05	82.80	81.66	85.25
AVG	**45.74**	37.90	**77.60**	64.70	**92.97**	77.81	47.90	49.63	60.94	65.20	65.11	73.11

Table 1: Results from Task 1 and Task 2. RNN refers to the RNN Encoder-Decoder with data augmentation and weighted voting presented in Section 4. Baseline refers to the non-neural baseline system presented in Cotterell et al. (2017). RNN accuracies which are greater than the baseline accuracy are shown in boldface.

guages.

The overall performance in Task 1 varies greatly between languages especially in the low and medium data settings. For example, the accuracy for Basque in the low setting is 4.00%, whereas the accuracy for Danish is 68.90%. One explaining factor may be the number of distinct morphological feature sets in the test data.

We found that there is a link between low accuracy and the number of distinct morphological feature sets occurring in the test data in the low training data setting, as is shown in Figure 5. A larger number of distinct feature sets correlates with lower accuracy. No such trend exists for the high or medium setting. This can partly be explained by the number of unseen morphological feature sets.

In languages with many different morphological feature sets, the test data may contain a large amount of morphological feature sets which were unseen in the low training data spanning 100 examples. This seems to adversely impact accuracy even though the Encoder LSTM does not treat morphological feature sets as atomic units (for example ”**V;PRS;PCP**”) but instead splits them into separate symbols (”**V**”, ”**PRS**”, ”**PCP**”). This conclusion is supported by the results for Basque: for the low setting, the system achieves accuracy 4%, whereas it achieves accuracy 100% for the high training data setting. A mere 8% of the morphological feature sets in the Basque test data occur in the low training data of 100 examples. However, 99% of them occur in the high training data containing 10,000 examples.

The present work employs a very naïve form of data augmentation. A new training example is created from an existing one by replacing the longest common substring of the stem and word form with a sequence of random characters from the training data. We also tried to use more sophisticated language models for generating the examples. Interestingly, this failed to bring improvements. In fact, it resulted in reduced performance. This may be due to overfitting because the generated strings too closely resemble existing training examples.

For eight languages (Dutch, Haida, Hungarian, Kurmanji, Latvian, Lithuanian, Navajo and Romanian), the RNN system failed to reach the baseline in the low training data setting. Except for Haida and Navajo, the difference between the baseline and the RNN system is, quite small ($\leq 5\%$). The

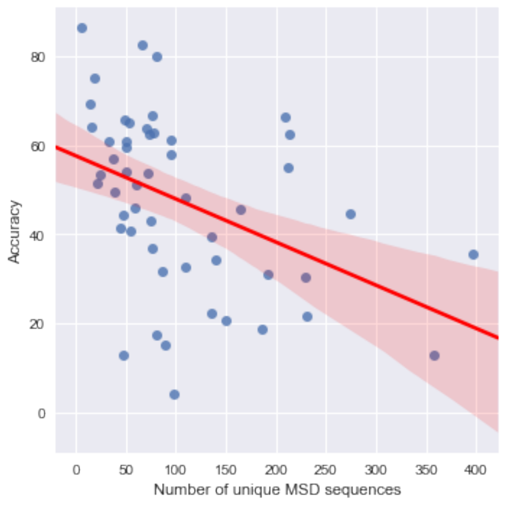

Figure 5: Low accuracy on the dev data (for the low setting in Task 1) trends downwards as the number of unique MSD combinations in a language's dev data increases. The red regression line shows the slope of this trend, with a 95% confidence interval represented as the translucent shadow around it.

Haida test set is very small (100 examples). Therefore, random fluctuations play a big role in the accuracy. For Navajo, the difference of 7.3%-points is substantial. We conjecture that this happens because data augmentation is not effective in the case of Navajo due to the short average length of the longest common substrings (LCS) of Navajo lemmas and word forms. For example, the average word lengths in the low training data for Navajo and Danish are nearly the same: 9.9 and 9.6 characters, respectively. However, the average length of the LCS of lemmas and word forms is a mere 2.9 characters for Navajo but it is 6.7 characters for Danish. Therefore, generated examples for Navajo will contain long substrings that occur in the original training data which may lead to overfitting.

In conclusion, we have demonstrated that an RNN Encoder-Decoder system can be applied to morphological reinflection even in a low resource setting. We achieve substantial improvements over a non-neural baseline in Task 1. However, the system performs poorly in Task 2 due to overfitting. Improving performance for Task 2 remains future work at the present time.

Acknowledgments

The third author has been partly sponsored by DARPA I20 in the program Low Resource Languages for Emergent Incidents (LORELEI) issued by DARPA/I20 under Contract No. HR0011-15-C-0113.

References

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *HLT-NAACL*.

Iñaki Alegria and Izaskun Etxeberria. 2016. EHU at the SIGMORPHON 2016 shared task. a simple proposal: Grapheme-to-phoneme for inflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*.

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531* .

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

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, Vancouver, Canada.

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 SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. *arXiv preprint arXiv:1705.00440* .

Felix A. Gers and Juergen Schmidhuber. 2000. Recurrent nets that time and count. Technical report. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale* .

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A search space odyssey. *CoRR* abs/1503.04069.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analyzing deep recurrent neural networks. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Curran Associates Inc., USA, NIPS'13, pages 190–198.

Katharina Kann and Hinrich Schütze. 2016. MED: The LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Lauri Karttunen and Kenneth R Beesley. 2005. Twenty-five years of finite-state morphology. *Inquiries Into Words, a Festschrift for Kimmo Koskenniemi on his 60th Birthday* pages 71–83.

David King. 2016. Evaluating sequence alignment for learning inflectional morphology. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*. Curran Associates Inc., USA, NIPS'12, pages 1097–1105.

Ling Liu and Lingshuang Jack Mao. 2016. Morphological reinflection with conditional random fields and unsupervised features. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. DyNet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980* .

Garrett Nicolai, Bradley Hauer, Adam St Arnaud, and Grzegorz Kondrak. 2016. Morphological reinflection via discriminative string transduction. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Robert Östling. 2016. Morphological reinflection with convolutional neural networks. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Alexey Sorokin. 2016. Using longest common subsequence and character models to predict word forms. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Dima Taji, Ramy Eskander, Nizar Habash, and Owen Rambow. 2016. The Columbia University - New York University Abu Dhabi SIGMORPHON 2016 morphological reinflection shared task submission. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics, Berlin, Germany.

Ian H. Witten and Timothy C. Bell. 1991. The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *Information Theory, IEEE Transactions on* (4):1085–1094.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710* .