ISI at the SIGMORPHON 2017 Shared Task on Morphological Reinflection

Abhisek Chakrabarty and **Utpal Garain**
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
203 B.T. Road, Kolkata-700108, India
abhisek0842@gmail.com, utpal@isical.ac.in

Abstract

We present a system for morphological reinflection based on the LSTM model. Given an input word and morphosyntactic descriptions, the problem is to classify the proper edit tree that, applied on the input word, produces the target form. The proposed method does not require human defined features and it is language independent also. Currently, we evaluate our system only for task 1 without using any external data. From the test set results, it is found that the proposed model beats the baseline on 15 out of the 52 languages in high resource scenario. But its performance is poor when the training set size is medium or low.

1 Introduction

The morphological reinflection task is to generate the variant of a source word, given the morphosyntactic descriptions of the target word. This year's shared task (Cotterell et al., 2017) is divided into two sub-tasks. Task 1 demands to inflect the isolated word forms based on labelled training data. For example, given the source form 'communicate' and the features 'V;3;SG;PRS', one has to predict the target form 'communicates'. Whereas, in task 2, partially filled incomplete paradigms are provided. The goal is to complete them using a restricted number of full paradigms. For each of the tasks, 3 separate training files are given per language, which differ in size (low/medium/high), in order to analyze systems' generalization ability in low and high resource situations. The competition is spread over 52 languages. For each language, a finite set of morphological tags are provided, from which the target inflections are taken. Evaluation is done separately under each of the three different training sets. To make the shared task competition fair, use of external resources are forbidden for the main competition track. However, for those systems which make use of external monolingual corpora, a list of approved external corpora selected from the Wikipedia text dumps are provided.

So far, there have been several efforts on reinflection employing statistical learning based methods (Dreyer and Eisner, 2011; Durrett and DeNero, 2013; Ahlberg et al., 2015; King, 2016) and string transduction (Nicolai et al., 2015). These methods entail feature definition which is hard to generalize for all of the world's languages.

In this article, we introduce a long short-term memory (LSTM) network architecture to handle the morphological reinflection task. The proposed method is language independent and does not require features to be defined manually. Our model is related to the encoder-decoder based approaches such as (Aharoni et al., 2016; Faruqui et al., 2016; Kann and Schütze, 2016a,b), but the main difference is that the proposed network is not designed to generate sequence of characters as output. Rather, we formulate the problem as to classify the transformation process required to convert a source form to its target form (Chakrabarty et al., 2017). Our goal is to model such a system which receives an input word and the morphological tags and returns the proper transformation that induces the target word. The source-target transformation is accomplished using edit tree (Chrupala et al., 2008; Müller et al., 2015). Initially all edit trees are extracted from the labelled pairs in the training data and then the distinct candidates from them are marked as the class labels. We feed the character sequence of the input word through the LSTM network to encode it and finally, the encoded representation is jointly trained with the input tags to classify the correct edit tree.

Currently, we assess our system only for task 1

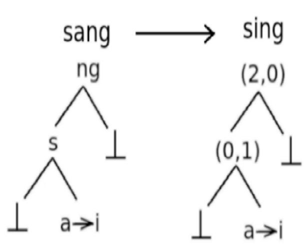

Figure 1: Edit tree for the source-target pair 'sang-sing'.

on all 52 languages, though it can be used for task 2 also. No external data such as the Wikipedia dumps provided by the SIGMORPHON committee has been exploited in the present work. The results obtained from the test sets indicate that the proposed method is resource intensive. When the training size is high, it achieves over the baseline system on 15 out of the 52 languages. But on medium and low amount training data, the performance is poor beating the baseline on 5 and 4 languages only.

2 Methodology

Edit Trees: An edit tree encodes a transformation which maps a source string to a target string. Given a source-target pair, the process of finding the corresponding edit tree is as follows. At first, the longest common substring (LCS) between them is found and then the prefix and suffix pairs of the LCS are recursively modelled in the same manner. The edit tree does not encode the LCS itself. Instead, it contains the length of the prefix and suffix in the source string for generalization. When no LCS is found between the source and the target strings, they are kept as a substitution node.

Figure 1 shows an example of edit trees between the source-target pair 'sang-sing'. The LCS between them is 'ng'. In the source string, the prefix length of the LCS is 2 (for 'sa') and the suffix length is 0. So, the root of the edit tree keeps the information $(2, 0)$. The left subtree of the root represents the edit tree between the prefix pair of the LCS in the source and the target string i.e. for 'sa-si' and following the same way, the right subtree remains empty.

Note that, to generalize the transformation pattern, the LCS is not stored in the edit tree. Consider the two source-target pairs 'gives-give' and 'takes-take' where the transformation rule is same i.e. to omit the ending 's' character. The root of the corresponding edit tree contains $(0, 1)$. If the LCS were stored in the root, then the tree could not be generalized for all the pairs like 'comes-come', 'sleeps-sleep' etc. where the same rule works.

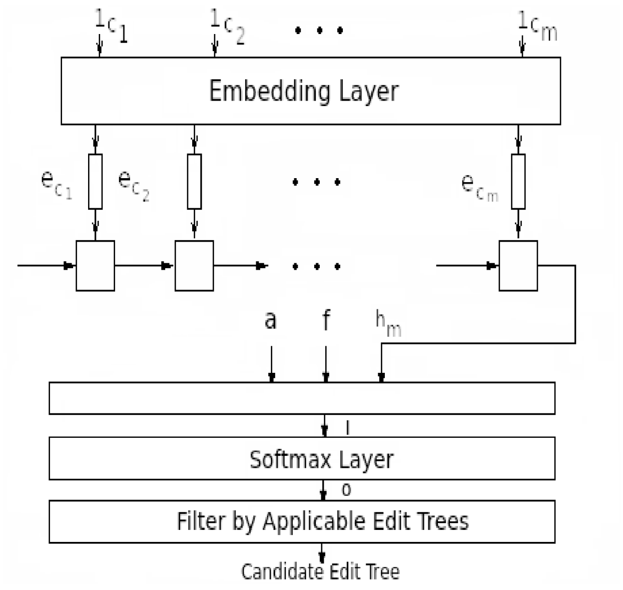

Figure 2: The model architecture.

2.1 The System Description

The architecture of our system is presented in Figure 2. At first, we use the LSTM network to make a syntactic embedding of the source word, that captures the morphological regularities. A character alphabet of the concerned language is defined as C. Let the input word w consists of the character sequence $c_1, \ldots, c_m$ where the word length is m and each character c_i is represented as a one hot encoded vector $\mathbf{1}_{c_i}$. The particular dimension of $\mathbf{1}_{c_i}$ referring to the index of c_i in the alphabet C, is set to one and the other dimensions are made zero. $\mathbf{1}_{c_1}, \ldots, \mathbf{1}_{c_m}$ are passed to an embedding layer $\mathbf{E}_c \in \mathbb{R}^{d_c \times |C|}$, which projects them to d_c dimensional vectors $\mathbf{e}_{c_1}, \ldots, \mathbf{e}_{c_m}$, by doing the operation $\mathbf{e}_{c_i} = \mathbf{E}_c \cdot \mathbf{1}_{c_i}$ where '$\cdot$' denotes matrix multiplication.

When the sequence of vectors $\mathbf{e}_{c_1}, \ldots, \mathbf{e}_{c_m}$ is given to the LSTM network, it computes the state sequence $\mathbf{h}_1, \ldots, \mathbf{h}_m$ using the following equations:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{e}_{c_t} + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{V}_f\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{e}_{c_t} + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{V}_i\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} \\
&\quad + \mathbf{i}_t \odot tanh(\mathbf{W}_c\mathbf{e}_{c_t} + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{e}_{c_t} + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{V}_o\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t),
\end{aligned}$$

σ denotes the sigmoid function and $\odot$ stands for the element-wise (Hadamard) product. LSTM utilizes an extra memory $\mathbf{c}_t$ that is controlled by three gates - input ($\mathbf{i}_t$), forget ($\mathbf{f}_t$) and output ($\mathbf{o}_t$). **W**, **U**, **V** (weights), **b** (bias) are the parameters. Eventually, we take the final state $\mathbf{h}_m$ as the encoded representation of w.

In addition to the source word, we have morphosyntactic features in hand to predict the target form. From the training data, all distinct features are sorted out to make a feature dictionary F. For a training sample, the given features are mapped to $|F|$ dimensional feature vector $\mathbf{f} = (f_1, \ldots, f_{|F|})$ where $f_i = 1$ if the i^{th} feature in the dictionary is present in the input features, otherwise f_i is set to 0. Thus, **f** becomes a numeric representation of the input features for the present training sample.

Another important point is that, for any arbitrary input word, all unique edit trees in the training data are not applicable due to incompatible substitutions. For example, the edit tree for the source-target pair 'sang-sing' (shown in Figure 1) cannot be applied on the word 'sleep'. In spite of all unique edit trees are set as the class labels, few of them are applicable for an input word to the model. To sort out this issue, we put the information over which classes the model should distribute the output probability mass while training.

Let $T = \{t_1, \ldots, t_k\}$ be the distinct edit trees set extracted from the training data. For the input word w, its applicable edit trees vector is defined as $\mathbf{a} = (a_1, \ldots, a_k)$ where $\forall j \in \{1, \ldots, k\}$, $a_j = 1$ if t_j is applicable for w, otherwise 0. Hence, **a** holds the applicable edit tree information for w. Finally, we combine the LSTM output $\mathbf{h}_m$, feature vector **f** and applicable tree vector **a** together for the edit tree classification task as following,

$$\mathbf{l} = softplus(\mathbf{L}^h\mathbf{h}_m + \mathbf{L}^f\mathbf{f} + \mathbf{L}^a\mathbf{a} + \mathbf{b}),$$

where 'softplus' is the activation function $f(x) = ln(1 + e^x)$ and $\mathbf{L}^h, \mathbf{L}^f, \mathbf{L}^a$ and **b** are the network parameters. Next, **l** is passed through the softmax layer to get the output labels for w.

To pick the maximum probable edit tree for an input word, we exploit the prior information about applicable classes. Let $\mathbf{o} = (o_1, \ldots, o_k)$ be the output of the softmax layer. The particular edit tree $t_j \in T$ is considered as the right candidate, where

$$j = \text{argmax}_{j' \in \{1,\ldots,k\} \,\wedge\, a_{j'}=1}\, o_{j'}$$

In this way, we choose the maximum probable class over the applicable classes only.

Language- Training Set Size	Our Model's Acc. (%)	Baseline Acc. (%)
albanian-high	79.1	78.9
arabic-high	60.2	50.7
armenian-high	89.5	87.2
dutch-high	90.2	87.0
georgian-high	94.4	93.8
hebrew-high	71.1	54.0
hindi-high	99.1	93.5
hungarian-high	77.5	68.5
icelandic-high	78.2	76.3
irish-high	60.6	53.0
italian-high	91.1	76.9
khaling-high	56.0	53.7
russian-high	86.1	85.7
turkish-high	73.5	72.6
urdu-high	98.2	96.5
english-medium	91.1	90.9
french-medium	73.9	72.5
hebrew-medium	46.6	37.5
italian-medium	75.7	71.6
scottish-gaelic-medium	62.0	48.0
albanian-low	21.6	21.1
danish-low	61.9	58.4
khaling-low	6.1	3.1
serbo-croatian-low	24.2	18.4

Table 1: Our model's performance on the test datasets for those languages where it beats the baseline system.

3 Experimentation

Parameters of the Model: For all 52 languages, we limit each word length to maximum 25 characters. Null characters are padded to the smaller words at the end and for words having more than 25 characters, extra characters are omitted. We represent each character as 25 length sequence of one hot encoded character vectors that are passed to the embedding layer. The output dimension of the embedding layer is set as the length of the one hot encoded character vectors i.e. $|C|$, size of the character alphabet of the concerned language.

Hyper parameters of the model are given as follows. The number of neurons in the hidden layer

Language- Training Set Size	Our Model's Acc. (%)	Baseline Acc. (%)
albanian-high	79.4	78.1
arabic-high	60	47.7
armenian-high	89.2	89.1
dutch-high	88	86.8
georgian-high	94.6	94
hebrew-high	72.3	55.8
hindi-high	99.1	94
hungarian-high	75.8	71.1
icelandic-high	80.6	76.1
irish-high	62.4	54.3
italian-high	91.6	79.9
khaling-high	56.2	53.8
russian-high	85.6	82
turkish-high	74.4	72.9
urdu-high	97.4	95.8
georgian-medium	90.4	90
hebrew-medium	47.3	40
italian-medium	78.2	73.8
scottish-gaelic-medium	68	52
danish-low	60.2	59.8
khaling-low	5	3.9
serbo-croatian-low	26.7	21.3
welsh-low	17	15

Table 2: Our model's performance on the development datasets for those languages where it beats the baseline system.

of LSTM is set to 64 for all languages. We apply online learning in our model. Number of epochs and the dropout rate are set to 150 and 0.2 respectively. We use 'Adagrad' (Duchi et al., 2011) optimization algorithm for training. Categorical cross-entropy function is used to measure the loss in our model.

3.1 Results

As stated in section 1, our method overperforms the baseline system on 15 out of the 52 languages in high resource configuration for the test sets. Whereas, in medium and low resource situations separately, it beats the baseline on 5 and 4 languages respectively. We provide these results in Table 1. The results show that the proposed method is resource intensive.

We also provide our model's performance on the development datasets in Table 2. The results are quite similar to the results given in Table 1. When the training size is high, the proposed model beats the baseline on 15 languages. For medium and low resource scenario, it achieves over the baseline on 4 languages only.

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, pages 41–48. http://anthology.aclweb.org/W16-2007.

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, Denver, Colorado, pages 1024–1029. http://www.aclweb.org/anthology/N15-1107.

Abhisek Chakrabarty, Onkar Arun Pandit, and Utpal Garain. 2017. Context sensitive lemmatization using two successive bidirectional gated recurrent networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada.

Grzegorz Chrupala, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with morfette. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA), Marrakech, Morocco. http://www.lrec-conf.org/proceedings/lrec2008/pdf/594$_{p}aper.pdf$.

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.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 616–627. http://www.aclweb.org/anthology/D11-1057.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159. http://www.jmlr.org/papers/v12/duchi11a.html.

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, pages 1185–1195. http://www.aclweb.org/anthology/N13-1138.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 634–643. http://www.aclweb.org/anthology/N16-1077.

Katharina Kann and Hinrich Schütze. 2016a. 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, pages 62–70. http://anthology.aclweb.org/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, Berlin, Germany, pages 555–560. http://anthology.aclweb.org/P16-2090.

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, pages 49–53. http://anthology.aclweb.org/W16-2008.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2268–2274. http://aclweb.org/anthology/D15-1272.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, pages 922–931. http://www.aclweb.org/anthology/N15-1093.