What can we gain from language models for morphological inflection?

Alexey Sorokin
Moscow Institute of Physics and Technology / Institutskij per., 9,
Faculty of Innovations and High Technologies, 141701, Dolgoprudny, Russia
Moscow State University / GSP-1, Leninskie Gory, 1
Faculty of Mathematics and Mechanics, 119991, Moscow, Russia
alexey.sorokin@list.ru

Abstract

This paper investigates the attempts to augment neural-based inflection models with character-based language models. We found that in most cases this slightly improves performance, however, the effect is marginal. We also propose another language-model based approach that can be used as a strong baseline in low-resource setting.

1 Introduction

Morphological inflection is a task of automatic reconstruction of surface word form given its source form, called lemma, and morphological characteristic of required form. For example, in Spanish the input word *contar* together with features v;fin;ind;imp;pst;3;pl should be transformed to *contaban*. The obvious way to solve such a task is to handcode transformations using finite-state rules. However, this approach requires an expert knowledge of the language under consideration and can be extremely time-consuming for the languages with complex morphology. Therefore a machine learning algorithm should be developed to efficiently solve this task for any language. Such an algorithm must be able to generalize from known lemma-features-word triples to previously unseen ones, mimicking human behaviour when inflecting a neologism in its native language or an unknown word in a foreign one. In this setting automatic inflection becomes an instance of string transduction problem, which makes conditional random fields a natural baseline model as suggested in (Nicolai et al., 2015). Another popular approach is to predict transformation pattern, either as a pair of prefix and suffix changes as used in the baseline model for Sigmorphon 2018 Shared Task (Cotterell et al., 2018). For example, consider a Czech adjective *krásný* and its superlative form *nejkrášnejšt́*. The inflection pattern can be encoded as a pair of prefix rule \$ → \$*nej* and a suffix rule *ý* → *ejšt́*. Such encoding is, however, too weak to deal with infixation and root vowel alterations, required, for example, for Spanish verb *volver* and its +Pres+Sg+1 form *vuelvo*. An *abstract paradigm* approach (Ahlberg et al., 2015; Sorokin, 2016) compresses this transformation to 1+o+2+er#1+ue+2+o, where digits stand for variables (the parts of verb stem), and constant fragments define the paradigm. In both cases in order to predict the inflected word form one suffices to guess the transformation pattern, thus solving a standard classification problem. Both mentioned models (CRFs and abstract paradigms) were quite successful in Sigmorphon 2016 Shared Task (Cotterell et al., 2016), however, they were clearly outperformed by neural network approaches.

Indeed, string transduction problems are successfully solved using neural methods, for example, in machine translation. The work of (Kann and Schütze, 2016) adopts the seq2seq model with soft attention of (Bahdanau et al., 2014). It defeated not only non-neural systems mentioned earlier, but also other neural approaches. It shows that attention mechanism is crucial for word inflection. However, in contrast to machine translation, a symbol of output word is less prone to depend from multiple input symbols, than a translated word from multiple source words. Consequently, the attention weight is usually concentrated on a single source symbol, being more a pointer than a distributed probability mass. Moreover, this pointer traverses the source word from left to right in order to generate the inflected form. All that motivated the hard attention model of (Aharoni and Goldberg, 2017), which outperformed the soft attention approaches. The key feature of this model is that it predicts not only the output word, but also the alignment between source and target using an additional step symbol which shifts the pointer to the

Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection, pages 99–104,
Brussels, Belgium, October 31, 2018.

next symbol. This model was further improved by (Makarov et al., 2017), whose system was the winner of Sigmorphon 2017 evaluation campaign (Cotterell et al., 2017). The approach of Makarov et al. was especially successful in low and medium resource setting, while in high resource setting it achieves an impressive accuracy of over 95%[1].

Does it mean that no further research is required and hard attention method equipped with copy mechanism is the final solution for automatic inflection problem? Actually, not, since the quality of the winning approach was much lower on medium (about 85%) and low (below 50%) datasets. This lower quality is easy to explain since in low resource setting the system might even see no examples of the required form[2] or observe just one or two inflection pairs which do not cover all possible paradigms for this particular form. For example, Russian verbs has several tens of variants to produce the +Pres+Sg+1 form. Consequently, to improve the inflection accuracy the system should extract more information from the whole language, not only the instances of the given form. This task is easier for agglutinative languages with regular inflection paradigm: to predict, say, the +Pres+Sg+1 form in Turkish, the system has just to observe several singular verb form (not necessarily of the first person) to extract the singular suffix and several first person form (of any number and tense). In presence of fusion, like in Russian and other Slavonic languages, the decomposition is not that easy or even impossible.

However, this decomposition is already realised in model of (Makarov et al., 2017) since the grammatical features are treated as a list of atomic elements, not as entire label. A new source of information about the whole language are the laws of its phonetics. For example, to detect the vowel in the suffix of the Turkish verb one do not need to observe any verbs at all, but to extract the vowel harmony patterns from the inflection of nouns. A natural way to capture the phonetic patterns are character language models. They were already applied to the problem of inflection in (Sorokin, 2016) and produced a strong boost over the baseline system. The work of Sorokin used simple ngram models, however, neural language models (Tran et al., 2016) has shown their superiority over earlier approaches for various tasks.

Summarizing, our approach was to enrich the model of (Makarov et al., 2017) with the language model component. We followed the architecture of (Gulcehre et al., 2017), whose approach is simply to concatenate the state of the neural decoder with the state of the neural language model before passing it to the output projection layer. We expected to improve performance especially in low and medium resource setting, however, our approach does not have clear advantages: our joint system is only slightly ahead the baseline system of (Makarov et al., 2017) for most of the languages. We conclude that the language model job is already executed by the decoder. However, given the vitality of language model approach in other areas of modern NLP (Peters et al., 2018), we describe our attempts in detail to give other researchers the ideas for future work in this direction.

2 Model structure

2.1 Baseline model

As the state-of-the-art baseline we choose the model of Makarov et al. (Makarov et al., 2017), the winner of previous Sigmorphon Shared Task. This system is based on earlier work of Aharoni and Goldberg (Aharoni and Goldberg, 2017). We briefly describe the structure of baseline model (we call it AGM-model further) and refer the reader to these two papers for more information. AGM-model consists of encoder and decoder, where an encoder is just a bidirectional LSTM. Each element of the input sequence contains a 0-1 encoding of a current letter and two LSTMs traverse this sequence in opposite directions. After encoding, each element of obtained sequence contains information about current letter and its context.

The main feature of the encoder is that it operates on the level on alignments, not on the level of letter sequences. Assume a pair *volver-vuelvo* appears in the training set. The natural alignment is

v	u	e	l	v	o	
|	|		|	|	|	
v	o		l	v	e	r

It is transformed to the source-target pair in Figure 1. Here the step symbol denotes pointer shift, for precise algorithm of transformation see (Aharoni and Goldberg, 2017):

The decoder is one-directional LSTM. It obtains

[1] Averaged over all languages of (Cotterell et al., 2017) dataset

[2] it was provided with 100 inflection pairs for entire language, which is often several times lower than the number of possible grammeme combinations

begin	v	step	u	e	step	l	step	v	step	o	step	step	end
begin	v	v	o	o	o	l	l	v	v	e	e	r	end

Figure 1: Transformation of alignment to source-target pair.

as input the lower string of Figure 1. Let i be the number of current timestep and j be current position in the input string. On i-th step the decoder takes a concatenation of 3 vectors: x_j — the j-th element in the output of the encoder, $\tilde{f} = W_{feat} f$ — the embedding of the grammatical feature vector and $g_i = W_{emb} y_{i-1}$ — the embedding of previous output symbol. The feature vector is obtained as $0/1$-encoding of the list of grammatical features. We actually take the concatenation of output vectors for $d \geq 1$ previous output symbols as y_{i-1}, in our experiments d was set to 4.

On each step the decoder produces a vector z_i as output and propagates updated hidden state vector h_i to the next timestep. z_i is then passed to a two-layer perceptron with ReLU activation on the intermediate layer and softmax activation on the output layer, which produces the output distribution p_i over output letters, formally:

$$\begin{aligned} \widehat{z}_i &= \max\left(W_p z_i + b_p, \overline{0}\right), \\ p_i &= \mathrm{softmax}(W_o \widehat{z}_i + b_o), \\ y_i &= \mathrm{argmax}_k\, p_{ik} \end{aligned}$$

If y_i is the index of step symbol, we move the pointer to the next input letter. We also use the copy gate from (Makarov et al., 2017): since the neural network copies the vast majority of its symbols, the output distribution $\widehat{p}_i$ is obtained as a weighted sum of singleton distribution which outputs current input symbol and the preliminary distribution p_i specified above. The weight σ_i is the output of another one-layer perceptron:

$$\begin{aligned} \sigma_i &= \mathrm{sigmoid}(W_\sigma z_i + b_\sigma), \\ \widehat{p}_i &= \sigma_i I(k = c_j) + (1 - \sigma_i) p_i, \\ y_i &= \mathrm{argmax}_k\, \widehat{p}_{ik} \end{aligned}$$

2.2 Character-based model

Our proposal is to explicitly equip the decoder with the information from the character-based language model. We suppose it will help the model to avoid outputting phonetically implausible sequences of letters. We choose the simplest possible architecture of the language model, namely, on each step it takes a concatenation of d previous symbol embeddings $u_i = [g_{i-d}, \ldots, g_{i-1}]$ and applies an LSTM cell to obtain a vector v_i and update LSTM hidden state h_i. v_i is propagated through a two-layer perceptron to predict the next output symbol analogously to the output layer of the baseline model:

$$\begin{aligned} \widehat{u}_i &= \max\left(W_p^{LM} u_i + b_p^{LM}, \overline{0}\right), \\ p_i^{LM} &= \mathrm{softmax}(W_o^{LM} \widehat{u}_i + b_o^{LM}), \\ y_i &= \mathrm{argmax}_k\, p_{ik}^{LM} \end{aligned}$$

The model is trained to predict next output symbol separately from the basic model. In principle, one can use more complex neural architectures, for example, a multilayer LSTM or apply attention mechanism. However, our preliminary experiments have shown that attention over recent history as in (Tran et al., 2016) leads to slightly worse performance.

To join the baseline model and the language model we concatenate the decoder output z_i with the analogous vector from the language model z_i^{LM}. The language model is conditioned over previously output vectors (excluding step symbol). That is the fusion mechanism as used in (Gulcehre et al., 2017). We also experimented with concatenating the pre-output vectors $\widehat{z}_i, \widehat{z}_i^{LM}$, however, the former variant leads to slightly better performance. To avoid exposure bias we mask language model state with all zeros with the probability of 0.4 (it teaches the model to recover from language model errors).

3 Data and implementation

3.1 Implementation

The initial alignment was obtained using longest common subsequence (LCS) method. Then this alignment was optimized using Chinese Restaurant process as in (Cotterell et al., 2016). The optimization phase did 5 passes over training data. The aligner trained on the training set was also used to align the validation data.

We implemented our model using Keras library with Tensorflow backend[3]. For all the setting we used the encoder with 96 hidden units in each direction, the decoder contained 128 units and the

[3] https://github.com/AlexeySorokin/Sigmorphon2018SharedTask

pre-output projection layer was of dimension 96. Morphological features were embedded to 48 dimensions. We used batch size of 32 when training, the batches contained the words of approximately the same size to reduce the amount of padding. We trained the model for 100 epochs with Adam optimizer, training was stopped when the accuracy on the validation data did not improve for 15 epochs. During decoding, the beam search of width 10 was applied.

When learning the weights of a language model, we used the same training and validation sets as for inflection network. The language model used history of 5 symbols and contained 64 units in LSTM layers. The number of layers was set to 2. The rate of dropout was the same as for basic model. The model was trained for 20 epochs, training was stopped when perplexity on validation set did not improve for 5 epochs.

3.2 Dataset

We tested our model in Sigmorphon 2018 Shared Task (Cotterell et al., 2018). For an extended description we refer the reader to this papers. The dataset contained three subsets: high, medium and low. The size of the training dataset was 10000 words in the high subset[4], 1000 in medium and 100 in low. The dataset also contained a development set containing 1000 instances most of the time, for all languages we used this subset as validation data. Overall, there were 86 languages in the high setting, 102 in medium and 103 in low.

4 Results and discussion

We submitted three systems, one replicating the algorithm of (Makarov et al., 2017), the second equipped with language models. The third one used only the language models: we extracted all possible abstract inflection paradigms for a given set of grammatical features and created a set of possible candidate forms applying all paradigms to the lemma. For example, consider the word *делать* and paradigms 1+ать#1+ет, 1+ать#1+ит, 1+ь#1 and 1+чь#1+жет; the first three produce the forms *делает*, *делит*, *делат*, while the fourth yields nothing since the given word does not end in *-чь*. Then all these forms are ranked using sum of logarithmic probabilities from forward and backward language models.

Our results are mostly negative, since our language-model based architecture produced only marginal improvement over the model of Makarov et al. which it is based on. Moreover, for the low-resource setting the performance of both system was mediocre, even our third paradigm-based system was able to overperform them despite its obvious weakness. The results are presented in Table 1[5], M1 stands for the baseline model and M2 – for the LM-based one. The numbers in brackets count the number of large gaps (more than 2% for high dataset, 3% for medium and 5% for low).

We observe that the influence of language models is marginal, the strength of this effect grows with the size of training data, which contradicts our expectations. In low and medium setting we expected slightly higher performance, which probably implies that our choice of hyperparameters is suboptimal. We made several observations when comparing our two models: first, the LM-based one demonstrates the highest quality after reducing output history of the baseline model from 4 to 2 and setting LM state dropout to 0.4. It shows that memory containing last output symbols plays the role of a language model for local dependencies and the memory of LSTM encoder – for global and often there is no need to duplicate them. However, most of the time LM-based variant converges much faster which implies that language model learns to throw out incorrect sequences of letters, but seems to overfit in the same time. In any case, these questions require future investigation.

However, language models demonstrate its utility even when little training data is available. The results for low subtask (see 2) demonstrate that they are powerful enough to discriminate between correct and incorrect variants proposed by the abstract paradigm generator. This is especially impressive since this method simply returns the input form in case it has not seen the given set of grammatical features. So it cannot recover the value of missing paradigm cells generalizing other elements of the paradigm table, which clearly limits its performance. Moreover, even for an observed grammatical values a small training set does not cover all possible inflection patterns, either due to their irregularity and multiplicity, like in case of Arabic or Latin, or complex phonetic rules as in case of Navajo. Nevertheless, this approach clearly

[4]For several languages it was smaller, but exceeded 1000 instances

[5]We measured accuracy on the test subset of the dataset and averaged the scores over all languages.

Dataset	M1	M2	$M1 > M2$	$M2 > M1$
high	94.23	94.56	27(2)	45(6)
medium	79.37	79.51	40(8)	47(12)
low	39.13	39.18	49(7)	50(10)

Table 1: Comparison of baseline and LM-equipped models

beats our neural models since it requires less data when the number of possible inflection patterns is small.

So language models are actually good in ranking inflection variants even in case of little data available. What remains is to generate enough candidate forms to improve their recall. We tried to solve this problem by adding top 10 candidates proposed by the neural network model to the list of possible outputs. However, this approach fails: for most languages the results fall below the level of neural models themselves. Doing a quick error analysis, we found that in low setting neural networks often are not able to discriminate between different forms, predicting a correct variant for another tense or person. The language model also does not learn enough well to distinguish different inflectional affixes due to the same lack of data. Therefore it favors either a shorter form or the endings it has observed more frequently, even if these endings does not refer to the set of features under consideration. On the contrary, abstract paradigms simply do not produce these variants, making the choice more easy. A possible workaround may be to predict the set of grammatical features for the generated form, however, we have not implemented this method due to the lack of time.

This reranking approach appears to be less successful for medium and high datasets. In this case the number of proposed candidate paradigms becomes too high. Some of these paradigms generate phonetically plausible forms but are applicable only in particular conditions not satisfied by a given word. For example, consider the Russian input *делать*;v;prs;ind;3;sg; the paradigm 1+ать#1+ит produces the form *делит*, which is correct, but for another verb *делить*. Therefore the application of language models in case of more training data looks problematic: we tried to use them to filter out forms generated by neural models without reranking remaining candidates. That marginally improved performance for complex languages like Navajo and Latin but had a slight negative effect in most other cases.

Acknowledgements

I thank the organizers of Sigmorphon2018 Shared task, especially Ryan Cotterell, for preparing the data and organizing the Shared Task. I am grateful to all the members of Neural Networks and Deep Learning Lab at MIPT (http://ipavlov.ai) for helpful discussions during my investigations. The research was conducted under support of National Technological Initiative Foundation and Sberbank of Russia. Project identifier 0000000007417F630002.

5 Conclusion

We investigated the applications of character language models to automatic reinflection. Despite their usefulness for other task, they do not produce significant boost, though improve the quality for all the settings. However, reranking-based approach, which also uses language models, reaches slightly higher scores in case of low amount of training data. In case of larger training sets the phonetic plausibility is effectively checked by the neural decoder itself without applying additional mechanisms. The relative success of paradigm-based approach in low-resource setting implies that neural networks lack control mechanism provided by abstract paradigms. Therefore the combination of neural networks with finite state techniques seems a perspective direction of study. Another promising direction not touched in the current work are different methods of data augmentation, either by training on data from related languages, or by generating additional training instances. At least for the second approach character language models seem useful to check the quality of generated source-target pairs.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 2004–2015.

Dataset	M1	M2	$M1 > M2$	$M2 > M1$
low	39.13	41.61	49(37)	52(44)

Table 2: Comparison of baseline and LM-paradigm models

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT 2015), Denver, CO*, pages 1024–1029.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sebastian Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, Brussels, Belgium. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Gėraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, et al. 2017. CoNLL–SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30.

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 2016 Meeting of SIGMORPHON*, Berlin, Germany. Association for Computational Linguistics.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, and Yoshua Bengio. 2017. On integrating a language model into neural machine translation. *Computer Speech & Language*, 45:137–148.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. *arXiv preprint arXiv:1606.00589*.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: Uzh at sigmorphon 2017 shared task for morphological reinflection. *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 49–57.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT 2015), Denver, CO*, pages 923–931.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

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*, pages 54–61.

Ke Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory networks for language modeling. *arXiv preprint arXiv:1601.01272*.