AX Semantics' Submission to the CoNLL-SIGMORPHON 2018 Shared Task

Andreas Madsack, Alessia Cavallo, Johanna Heininger and **Robert Weißgraeber**
AX Semantics, Stuttgart, Germany
{firstname.lastname}@ax-semantics.com

Abstract

This paper describes the AX Semantics submission to the SIGMORPHON 2018 shared task on morphological reinflection.

We implemented two systems, both solving the task for all languages in one codebase, without any underlying language specific features. The first one is a classifier, that chooses the best paradigms to inflect the lemma; the second system is a neural sequence model trained to generate a sequence of copy, insert and delete actions. Both provide reasonably strong scores on solving the task in nearly all 103 languages.

1 Introduction

This paper describes our implementation and results for Task 1 of the CoNLL-SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection (Cotterell et al., 2018). The task is to generate inflected word forms given the lemma and a feature specification (Kirov et al., 2018). See Figure 1 for an example in German.

`sehen` (V;IND;PST;3;PL) $\rightarrow$ `sahen`

Figure 1: Task 1 Example, German: putting the verb "sehen" into 3rd person past tense.

Including the surprise languages the task consists of 103 languages. Three differently-sized training sets were made available, namely a low dataset containing only 100 samples, a medium set with 1000, and a high set with 10000 samples. Most of the languages had all three sizes (89 languages), some only low and medium (13 languages), and one language only low.

We tackled the problem with two very different approaches. System 1 is training a classifier to predict an abstract paradigm according to which the inflected form is created and the system 2 is a character-based recurrent neural network.

2 System 1 - Paradigm classification

Our first approach to solve the task is based on the system proposed by Sorokin (2016), which participated in the SIGMORPHON-2016 Shared Task (Cotterell et al., 2016). It realizes the automatic inflection of word forms via the classification of abstract paradigms created through the method of longest common subsequence (LCS).

The idea of abstract paradigms was introduced in Ahlberg et al. (2014) and implies the representation of a lemma and an inflected form as a list of patterns, where common parts of both words are replaced by variables. Given e.g. the lemma *write* and the target form *writing* the abstract paradigm is **1+e#1+ing**. In order to create such a paradigm the LCS of both words has to be determined, which later will be replaced by variables. Such a representation suggests that the LCS is the stem of both words and the symbols not in the LCS characterize the inflection.

Unlike Sorokin (2016) and Ahlberg et al. (2014), who applied a finite state machine to extract the LCS and Ahlberg et al. (2015) who used an SVM classifier, we made use of a sequence alignment function provided by Biopython[1]. With the input parameters *write* and *writing* the method returns the alignment shown in Figure 2, based on which the LCS and later the paradigm can be constructed. Another system searching for best edit pairs is Morfette (Grzegorz Chrupala and van Genabith, 2008), which in contrast to our approach first reverses the strings to find the shortest sequence of insert and delete commands instead of the LCS.

If multiple alignments and hence LCS exist, we rate the options based on a set of rules and thereafter choose the alignment with the minimum score. More specifically, 0.5 points are given for a

[1]https://github.com/biopython/biopython

```
write__
||||···
writing
```

Figure 2: Formatted output of the pairwise2 function in Biopython for the strings *write* and *writing*

gap (specified by _) in one of both words, 1 point is given for each unequal character in the alignment (specified by .), and an additional amount of 100 is added to the score if the alignment creates following variables in the part of the abstract paradigm representing the lemma. The latter part is especially important as the value of subsequent variables, the common part in both words they stand for, can't be determined unambiguously. This poses a problem when the target form has to be constructed based on the variables in the lemma paradigm. We will discuss this issue in more detail in the last part of this section.

Following the procedure described above we created an abstract paradigm for each lemma and inflected form provided in the maximum available training set (if no high set was available we used the medium or low one respectively) for each language and predicted the corresponding proper test data. Thereupon we could train a classifier to predict the correct abstract paradigm given a lemma and the morphosyntactic description (e.g. `N;NOM;PL`) of its inflected form. Apart from the morphosyntactic description (MSD), we used 3 prefixes, as well as 5 suffixes of the lemma as input features for the classification. We applied one-hot-encoding on the features, creating a sparse matrix consisting of only 0s and 1s (Table 1) and eliminated all pre- and suffixes that occurred less than three times in the lemmas of the training set. This type of feature selection is the main difference between our system and the one described in Sorokin (2016), which apart from excluding the features seen less than 3 times only kept 10% of all features according to an ambiguity measure.

en	**sch**	**rite**	**ite**	**e**	**..**	**PST**	**NOM**
1	0	0	0	1	..	1	0
0	0	1	1	0	..	0	1
..	..	..	..	..	..	..	..

Table 1: Abstract illustration of a feature matrix used for the classification

In order to find the best performing classifier we inspected several algorithms available in the sklearn library for Python and conducted a randomized search for the best hyperparameters of each classifier. For 90 out of 103 languages a neural network yielded the best results on the development-set followed by the Decision Tree (8 languages), Support Vector Machines (3 languages), Random Forest (1 language), and Logistic Regression (1 language) algorithm.

After the classification of an abstract paradigm for a lemma and a MSD, the only task left is constructing the inflected form based on the abstract paradigm and the lemma. The basic procedure for the generation is to first identify the value of the variables in the abstract paradigm and then to insert these letter sequences into the abstract pardigm representing the inflected target form. However, as previously indicated, this procedure does not always deliver an unambiguous result. For example for the German lemma *sehen* (*to see*) and the MSD `V;IND;PST;3;PL`, which would be the target form *sahen* (*they saw*), the classifier should correctly predict the abstract paradigm **1+e+2#1+a+2**. Now there are two fitting value combinations, which would reconstruct the lemma, namely `1=s, 2=hen` and `1=seh, 2=n`, and hence two possible target forms exist (*sahen* and *sehan*), only one of which is the right target. The depicted example is fairly simple, but more complex samples, especially when the lemma paradigm consists of subsequent variables, could produce an even larger number of possible targets. The fraction of samples that yields more than one combination and the number of combinations if this is the case depends heavily on the language of interest. English e.g. has only one possible target form for 995 out of 1000 test samples and two possibilities for the remaining five, whereas Arabic produces a much more complicated result (1 combination = 5698 times, 2 combinations = 2677 times, 3 combinations = 1205 times, 4 combinations = 267 times, 5 combinations = 43 times, 6 combinations = 1 times).

To address the problem of multiple possible combinations we constructed a set of decision hierarchy based on which one combination is chosen. For each sample in the training set we recorded the value combination that led to the inflected form. A combination was coded as the index of the fitting value of a variable in the list of all possible values sorted by length. Then we could identify the combination that most frequently led

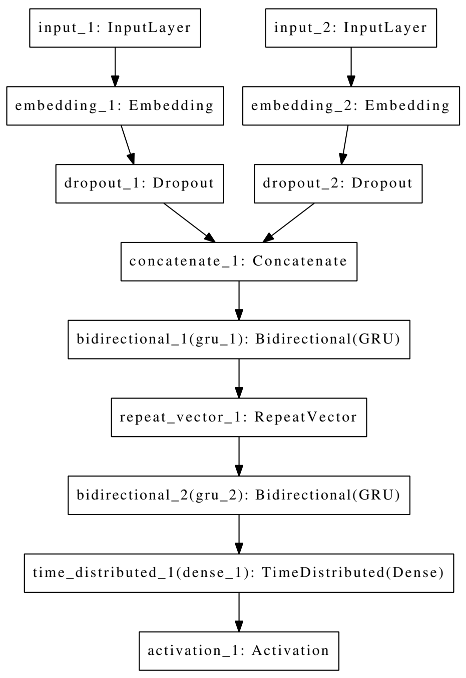
input_1: InputLayer
input_2: InputLayer
embedding_1: Embedding
embedding_2: Embedding
dropout_1: Dropout
dropout_2: Dropout
concatenate_1: Concatenate
bidirectional_1(gru_1): Bidirectional(GRU)
repeat_vector_1: RepeatVector
bidirectional_2(gru_2): Bidirectional(GRU)
time_distributed_1(dense_1): TimeDistributed(Dense)
activation_1: Activation

Figure 3: Sequence model for System 2

to the correct target for a specific set of variables with a specific number of possible values for each variable. In the example above we have two variables, `1 = [s, she]` and `2 = [hen, n]`, with two possible values each. For this specific case we can now look up the index of the values that most often led to a correct target during training and choose the combination which consists of these values for the generation of the inflected form.

3 System 2 - Sequence Neural Model

Our second system is based on the paper by (Makarov et al., 2017) which participated in the SIGMORPHON2017 Shared Task (Cotterell et al., 2017).

We used hamming distance (similar to the baseline code given by the organizers) to align the lemma and the expected result. On this alignment we generated `Copy`, `Delete` and `Insert` operations to transfer the lemma to the inflected form (analogous to Makarov et al. (2017, chaper 4.1)). For each language we trained a different model with a different charmap only consisting of the characters in the given language. This charmap is used for Insert operations. The sequence model is implemented using keras and Tensorflow. See model overview in Figure 3. The character based lemma input and the feature matrix are both encoded in their own embeddings. The feature matrix is a list of all possible features in all languages. This list is the same as the one for System 1.

The hyperparameters of the model are a dropout of 0.2, the number of features is 370, lemma input and output lengths are 100, and length of feature sequence is set to 20. The 2 bidirectional GRUs (Chung et al., 2015) have 256 hidden units. The activation function used is softmax as the output is a sequence of `Stop`, `Copy`, `Delete` and `Insert` of a character from the charmap. Every time the full sequence accuracy improved the model is saved. The full sequence accuracy is defined by the fact that all characters in a sequence are correct. The optimizer used is adam (Kingma and Ba, 2014).

4 Results

The performance of the presented systems on the test data is shown in Table 2 and Table 3 (on page 4). It can be seen that the first system yields slightly better results compared to the second system. The paradigm model outperforms the baseline in 77 cases whereas the sequence model beats the baseline in 53 out of 103 languages. Overall, both systems do fairly well compared to the baseline in nearly all languages, whereas when the baseline comes close the difference is only a minor fractions of the accuracy percentages. Regarding the paradigm model it has to be stated that the displayed values are probably lower than the classification accuracy, meaning that in some cases the correct paradigm may have been predicted, but the right target could not be constructed. We can't verify this assumption for the test data, but on the development data we observed that for some languages the classification performance was a lot better than the final accuracy. Irish e.g. had a classification accuracy of nearly 78% for the abstract paradigms, but the final performance amounted only to ca. 68% due to the mistakes made during the target creation. Improving the method of handling multiple possible targets could therefore further enhance the performance of the paradigm model.

Unsurprisingly, languages that provided lower numbers of data size don't perform very well overall.

For comparison of the error intensity of the two systems we calculated the Levenshtein distance between the system results and the expected in-

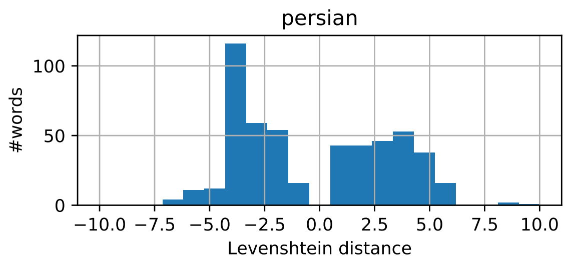

Figure 4: Levenshtein distance histogram for Persian. Errors left for system 1; right for system 2. Distances without error are removed.

language	data size	system 1	system 2	baseline
cornish	medium	2	**32**	12
greenlandic	medium	54	**82**	72
karelian	medium	62	**98**	42
kashubian	medium	**76**	60	68
kazakh	medium	44	**86**	50
khakas	medium	**96**	**96**	84
mapudungun	medium	90	**98**	82
middle-high-german	medium	**84**	52	54
middle-low-german	medium	**76**	30	38
murrinhpatha	medium	**54**	4	0
norman	medium	28	34	**46**
old-irish	medium	8	8	**16**
scottish-gaelic	medium	**70**	50	50
telugu	low	**72**	46	70
tibetan	medium	**44**	42	36
turkmen	medium	**96**	94	68

Table 2: Results for test data compared to baseline (medium and low)

flected forms. Of course if the system has a better accuracy on the task, it makes fewer errors than the other system. This calculation is not very conclusive, but allows for some basic predicates, see Figure 4 for a histogram of error distributions. When both systems are equally strong the sums of the distances are not that divergent. For example for *Persian* the sum of Levenshtein distances for system 1 is 919 and for system 2 it is 804 but for *Georgian* the distance for system 1 is 40 and for system 2 it is 55.

5 Conclusion

Our goal is to improve our morphology system component in our Natural Language Generation SaaS (Weißgraeber and Madsack, 2017). The two systems described herewithin compete against a handcrafted morphology and a reasonable lexicon. The handcrafted morphology and the lexicon is always better on very regular POS types (i.e. German adjectives). So all three systems (the two described in this paper and the handcrafted one) are evaluated for every language and POS type, and can be combined into a best-of-breed selection scheme by preferring the most appropriate system for each POS type and language combination.

language	data size	system 1	system 2	baseline
adyghe	high	99.0	**99.6**	91.6
albanian	high	**88.9**	40.9	79.5
arabic	high	**58.2**	34.0	4.1
armenian	high	**90.3**	73.4	86.6
asturian	high	**95.3**	91.1	95.2
azeri	high	94.0	**100.0**	70.0
bashkir	high	**99.8**	**99.8**	90.7
basque	high	**8.2**	5.4	7.3
belarusian	high	47.3	**52.8**	41.0
bengali	high	**96.0**	78.0	81.0
breton	high	80.0	**85.0**	73.0
bulgarian	high	88.7	75.6	**89.0**
catalan	high	**95.9**	92.1	95.7
classical-syriac	high	**100.0**	99.0	97.0
crimean-tatar	high	**98.0**	**98.0**	95.0
czech	high	89.2	83.0	**90.6**
danish	high	**92.7**	90.4	87.0
dutch	high	85.1	**88.6**	87.7
english	high	95.8	**96.5**	95.9
estonian	high	**87.7**	64.1	78.0
faroese	high	**79.6**	76.8	76.1
finnish	high	77.0	52.0	**78.0**
french	high	**85.3**	79.6	83.0
friulian	high	97.0	**99.0**	96.0
galician	high	**96.7**	95.0	95.1
georgian	high	95.1	**95.4**	93.9
german	high	**82.3**	**82.3**	81.1
greek	high	78.2	54.8	**78.3**
haida	high	93.0	**100.0**	66.0
hebrew	high	**84.3**	54.5	53.7
hindi	high	**100.0**	80.0	93.0
hungarian	high	76.9	**80.9**	68.8
icelandic	high	**80.9**	79.3	76.9
ingrian	high	44.0	**80.0**	46.0
irish	high	**67.2**	34.1	53.0
italian	high	**94.2**	63.7	77.5
kabardian	high	**99.0**	**99.0**	86.0
kannada	high	90.0	**97.0**	66.0
khaling	high	**72.0**	17.1	53.7
kurmanji	high	92.6	87.8	**92.9**
ladin	high	**93.0**	87.0	92.0
latin	high	46.2	37.2	**47.6**
latvian	high	**93.2**	90.2	92.8
lithuanian	high	**70.6**	52.0	64.2
livonian	high	**82.0**	76.0	67.0
lower-sorbian	high	94.2	**95.5**	88.1
macedonian	high	92.7	**94.2**	91.2
maltese	high	**63.0**	28.0	16.0
middle-french	high	**97.0**	95.4	95.1
navajo	high	**43.6**	6.8	0.0
neapolitan	high	94.0	**95.0**	**95.0**
northern-sami	high	61.7	**75.5**	62.3
north-frisian	high	**80.0**	33.0	37.0
norwegian-bokmaal	high	90.8	87.2	**91.0**
norwegian-nynorsk	high	82.8	**88.0**	74.8
occitan	high	94.0	92.0	**96.0**
old-armenian	high	**84.9**	82.2	79.2
old-church-slavonic	high	**92.0**	88.0	80.0
old-english	high	**69.3**	34.3	40.9
old-french	high	80.8	**82.0**	80.7
old-saxon	high	**87.3**	54.0	60.1
pashto	high	**92.0**	78.0	72.0
persian	high	63.7	62.6	**80.7**
polish	high	**87.6**	82.9	87.1
portuguese	high	**97.3**	94.6	96.7
quechua	high	**99.8**	98.8	95.1
romanian	high	**82.6**	62.4	79.8
russian	high	**88.0**	76.1	86.5
sanskrit	high	92.8	**93.7**	80.6
serbo-croatian	high	**87.4**	69.1	83.0
slovak	high	**91.5**	91.1	83.1
slovene	high	7.0	**90.9**	79.7
sorani	high	**76.0**	27.6	63.6
spanish	high	**94.4**	81.3	92.4
swahili	high	**98.0**	1.0	0.0
swedish	high	88.0	**88.5**	84.7
tatar	high	**99.0**	97.0	95.0
turkish	high	87.9	**95.9**	73.2
ukrainian	high	**93.1**	87.2	86.3
urdu	high	**99.3**	83.7	95.9
uzbek	high	**100.0**	99.0	96.0
venetian	high	**98.5**	97.6	93.0
votic	high	37.0	**66.0**	34.0
welsh	high	82.0	**88.0**	72.0
west-frisian	high	**82.0**	67.0	67.0
yiddish	high	97.0	**99.0**	94.0
zulu	high	**93.8**	2.5	0.2

Table 3: Results for test data compared to baseline (high)

Of course, all three systems have pros and cons. The handcrafted one fails on completely new words (or even rare words), that are not regularly inflected. The paradigm system is in some languages better compared to the sequence model and the errors of the paradigm system are not that disturbing, since they are usually more plausible, whereas the sequence model tends to make more arbitrary errors.

On error the sequence model may return for example something like this: *gerksent* for *murksen* `V.PTCP;PST` (German, correct form: *gemurkst*). An examplary major error for the paradigm system would be *kiefen* for *kaufen* `V;PST;3;PL` (correct form: *kauften*), where a native speaker can see the relation to the inflection of *laufen*, where the past form is *liefen*. This kind of errors are greatly reduced by training with *a lot* more data.

In the future we will try to improve especially the sequence model for the languages we use on our platform.

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2014. Semi-supervised learning of morphological paradigms and lexicons. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 569–578. Association for Computational Linguistics.

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*, pages 1024–1029, Denver, Colorado. Association for Computational Linguistics.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. *CoRR*, abs/1502.02367.

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

GD Grzegorz Chrupala and J van Genabith. 2008. Learning morphology with morfette. In *Proceedings of LREC*, volume 8.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Christo Kirov, Ryan Cotterell, John Sylak-Glassman, Graldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sebastian Mielke, Arya D. McCarthy, Sandra Kbler, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. UniMorph 2.0: Universal Morphology. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

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

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. Association for Computational Linguistics.

Robert Weißgraeber and Andreas Madsack. 2017. A working, non-trivial, topically indifferent nlg system for 17 languages. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 156–157. Association for Computational Linguistics.