KU-CST at the SIGMORPHON 2020 Task 2 on Unsupervised Morphological Paradigm Completion

Manex Agirrezabal and **Jürgen Wedekind**
Centre for Language Technology (CST)
University of Copenhagen
manex.aguirrezabal@hum.ku.dk, jwedekind@hum.ku.dk

Abstract

We present a model for the unsupervised discovery of morphological paradigms. The goal of this model is to induce morphological paradigms from the bible (raw text) and a list of lemmas. We have created a model that splits each lemma in a stem and a suffix, and then we try to create a plausible suffix list by considering lemma pairs. Our model was not able to outperform the official baseline, and there is still room for improvement, but we believe that the ideas presented here are worth considering.

1 Introduction

In this paper we describe our attempt to capture morphological paradigms totally from scratch (Kann et al., 2020) prepared for the task of morphological paradigm completion in the CoNLL–SIGMORPHON 2020 Shared Task. Computational morphology is not a new area and there is plenty of related work. Some years ago, this problem was commonly tackled using finite-state and two-level approaches, such as in Kaplan and Kay (1994), Beesley and Karttunen (2003), and Koskenniemi (1983). Recent works, on the other hand, rely mostly on statistical approaches, such as in Faruqui et al. (2016) and Kann and Schütze (2017).

There have been several Shared Tasks recently on morphological inflection (Cotterell et al., 2016, 2017, 2018; McCarthy et al., 2019). The task for this year is more complex, as we are asked to discover paradigms from scratch. This is an intriguing research area that could give us the chance of recovering dead languages that have only limited written resources. Several researchers have attempted to solve this task, such as Goldsmith et al. (2017), Jin et al. (2020), and Erdmann et al. (2020).

We present a pipeline that assumes that all morphological realizations in a paradigm (for each language) follow a fixed structure: `stem+suffix`. Based on that logic, we look for the best candidates to compose the suffix inventory, we cluster them using K-means and after that, we join stems and suffixes. We employ language models to get the most natural outputs. The pipeline that we have developed does not contain any neural network component, but we contemplate it as a possibility to extend our work in the near future.

This paper is structured as follows: In the next section we introduce the task that we have worked on. We describe our approach in the third section. Afterwards, we show our results compared to the baseline model. To conclude, we discuss our results and provide possible future directions.

2 Task

In this competition there was one task that we had to perform. A computational system had to be built, which, given a raw text and a set of lemmas, it would return the complete list of paradigms for each verb. The computational model should be able to read a text like this,

> *The aircraft landed at the JFK airport. Other pilots decided to land in Philadelphia. As you may imagine, landing a plane is not an easy job, but imagination can help.*

and extract morphological paradigms. In the shared task, a list of lemmas is also given as a starting point. This list of lemmas could include verbs like *land* and *imagine*.

In the case of the verb *land*, in the example above, it is pretty easy to get its inflections (*land*, *landed*, *landing*). This could, for example, be done with a Minimum Edit Distance based method and it is relatively easy, as there is no usage of *land* with the function of a noun. It gets slightly more complicated with the verb *imagine*, as a simple distance-based algorithm could fail, because it could find *imagination* as a possible conjugation of the verb *imagine*.

2.1 Dataset

As one of the most widely extended resources is the bible, the organizers decided to consider it as the raw text input data. Together with the bible, a list of verbal lemmas was given. The languages for development were Maltese, Persian, Portuguese, Russian and Swedish. The languages for testing included the following: Basque, Bulgarian, English, Finnish, German, Kannada, Navajo, Spanish and Turkish.

3 Method

Our method has a very strong assumption, which oversimplifies the problem but it also gives the chance of recognizing some patterns. The assumption is that all lemmas and their inflections have the following form for all languages

$$STEM{+}SUFFIX \rightarrow STEM{+}SUFFIX'$$

as illustrated in the following examples for English and Spanish:

$$\begin{array}{l|l} play \rightarrow playing & play{+}\epsilon \rightarrow play{+}ing \\ jugar \rightarrow jugando & jug{+}ar \rightarrow jug{+}ando \end{array}$$

Pipeline

We use a pipeline that includes four different steps. These are described below.

3.1 Step 1

In the first step, for each lemma l in the lemma list L and each word w in the corpus/dictionary D, all possible splits $l_1^1{+}l_2^{|l|}$, $l_1^2{+}l_3^{|l|}$,.., $l_1^{|l|}{+}\epsilon$, and $w_1^1{+}w_2^{|w|}$, $w_1^2{+}w_3^{|w|}$,.., $w_1^{|w|}{+}\epsilon$ are generated. (We use v_i^j, with $0 \leq i \leq j \leq |v|$, to denote the substring $v_i..v_j$ of a string v.) We assume the stem (the hypothesized *STEM*) to be nonempty but allow the suffix to be empty. For the Spanish lemma *jugar* we thus get $j{+}ugar$, $ju{+}gar$, $jug{+}ar$, $juga{+}r$, and $jugar{+}\epsilon$.

3.2 Step 2

In the second step, we determine the inflections of the regular verbs of the language. These will be used for the estimation of the morphological richness r_m of the lemmas (verbs) in the third step. The morphological richness of the lemmas can be identified with the number of combinations of those tense, aspect, mood, and agreement features that can be distinctively morphologically realized. Because the morphological richness of the lemmas (verbs) does not tend to vary much across the different lemmas (verbs), even if they inflect semi-irregularly or irregularly, we assume that each lemma has r_m different inflections. r_m thus provides an upper bound on the number of cells of the paradigms of the language/corpus.

For determining r_m we identify the inflections of the lemmas with regular inflection. First, we determine for each splitted lemma $l = r{+}s$ the number of potential inflections of the hypothesized stem r, that is $r{+}s'$, in D. This is the set $S_{r+s} = \{s' \mid r{+}s' \in D\}$. Then, for regularly inflecting lemmas, S_{r+s} will be large for the actual split but also for any split within the stem. This is illustrated for the German lemma *spielen* (play) with the actual split *spiel*$+$*en* below.

$S_{\textit{spiele+n}} = \{\epsilon, n\}$
$S_{\textit{spiel+en}} = \{e, st, t, en, ...\}$
$S_{\textit{spie+len}} = \{le, lst, lt, len, ...\}$

To accommodate for this deficiency, we also consider pairs of splitted lemmas $l = r{+}s$, $l' = r'{+}s$ with distinct stem endings $r_{|r|} \neq r'_{|r'|}$ and we determine the split $\hat{i}$ of s that yields the maximum number of common inflections:

$$\hat{i} = \max\nolimits_{i \in \{0,..,|s|\}} |S_{rs_0^i + s_{i+1}^{|s|}} \cap S_{r's_0^{i-1} + s_i^{|s|}}|$$

We choose for each lemma pair l, l' the splits $\hat{r}{+}\hat{s}$ and $\hat{r}'{+}\hat{s}$, with $\hat{r} = rs_0^{\hat{i}}$, $\hat{r}' = r's_0^{\hat{i}}$, and $\hat{s} = s_{\hat{i}+1}^{|s|}$, and consider their common suffixes in D: $S_{\hat{r}+\hat{s}} \cap S_{\hat{r}'+\hat{s}}$.

Because regularly inflecting verbs tend to share their inflections, this lemma pairing allows us to reliably predict that, for example, the stems of *spielen* and *gehen* are *spiel* and *geh*.

$S_{\textit{spiele+n}} \cap S_{\textit{gehe+n}} = \{\epsilon, n\}$
$S_{\textit{spiel+en}} \cap S_{\textit{geh+en}} = \{e, st, t, en, ...\}$
$S_{\textit{spie+len}} \cap S_{\textit{ge+hen}} = \emptyset$

Finally, for all splitted lemmas $\hat{r}{+}\hat{s}$ we collect the suffixes in $S_{\hat{r}+\hat{s}}$ in one bag.

3.3 Step 3

The goal of this step is to group different realizations of the same suffix. The previous step captures relevant suffixes, but in some cases, some parts of the stem are also included in these suffixes, or there might be some slight differences, because of morphophonological changes. In order to group them, we employ K-Means.

When using K-means we need a function that calculates the distance between the elements, and based on this distance, the instances will be clustered. We decided to employ a modified version of Minimum Edit Distance. Our modified version tries to punish changes that are made at the end of the suffix. The assumption in this case, is that changes at the beginning of the suffix are more likely to be caused by the stem (and they could be the same suffix). On the other hand, if there are changes at the end, it would be a different suffix. Our edit distance algorithm allows insertion and deletion as possible changes. We also assume that it is worse to substitute a vowel with a consonant, than changing a vowel with a vowel. Therefore, this would happen:

Distance (*era*, *bra*) $>$ Distance (*era*, *ara*)

	ntar	*ntaron*	*aron*	*ar*
ntar	0.000	0.939	0.778	0.094
ntaron	-	0.000	0.015	0.832
aron	-	-	0.000	0.656
ar	-	-	-	0.000

We estimate that the number of paradigms (r_m) in a language is approximately the third of the number of different suffixes found in the previous step. This number was estimated based on the behaviour of the model considering Swedish data. Therefore, K-means will reduce the number of possible suffixes to the third (this is a parameter that will be tuned in the future). For example, one of the clustered groups found in this step considering the Spanish data would be this: $\{$*rá*, *erá*, *derá*, *ará*, *irá*$\}$. This corresponds to the suffix of future simple, third person singular.

3.4 Step 4

In the previous steps we will have generated possible suffixes for each cell in a paradigm. Now, the goal is to make a guess of how a word form should be generated. For example, in Spanish, if we have the lemma *sanar*, and we want to build the first person singular of the future simple tense (*sanaré*), we could expect the lemma to be combined with suffixes like *é*, *ré*, *aré*, *iré*, and so on. These suffixes would be the output of the previous step.

First of all, for each lemma, the model needs to decide the position in which we will split the lemma, as following the previous assumption a word will have this shape: *STEM+SUFFIX*. In order make that decision, we check how often we associate each lemma with a specific stem in the output of step 2, and use the most frequently occurring stem for all the suffixes. For example, for the verb *sanar*, in Spanish, we get these frequencies: `san:15, sana:21, sa:1`, and therefore, we would use the stem *sana*.

We, then, try to join that stem with the clustered suffixes. Each stem will be joined with one suffix from each cluster. In order to decide which is the best suffix, we use a bigram character-level language model to estimate the probability of the output sequences, trained on the input bible. These are the probabilities that we get if we consider the example of the stem *sana* (from *sanar*) and suffixes *é*, *ré*, *aré* and *iré* in Spanish.

Candidate output	Probability
sanaé	0.0
sanaré	$4.097e-07$
sanaaré	$1.272e-10$
sanairé	$2.201e-10$

Obviously, in this case, the conjugation *sanaré* would be returned.

3.5 Expansion of the lemma list

At this point, the model produced a little amount of suffixes. Then, we decided to extend the list of input lemmas, so that it can find new suffixes and increase, therefore, the recall of the model.

We obtain new lemmas by training a very simple verb classifier. We create a simple dataset with the input lemmas and some random words from the corpus. The input lemmas will be tagged as verbs and the random words will be tagged as non-verbs. We, then, train a simple Logistic Regression model, using character uni-, bi- and trigrams for representing each word. We also include word boundary symbols. For instance, in Spanish we would have cases like:

Word	Features (trigrams)	class
comer	<`co`, `com`, `ome`, `mer`, `me`>	`V`
plaza	<`pl`, `pla`, `laz`, `aza`, `za`>	`NV`

Using this approach we obtain new verbs that can be used in our Pipeline. The model that uses the extended list of lemmas for extracting suffixes is called the **Flexible model**, and on the other hand, the initial model (the one that uses only the initial lemmas as input) is called the **Non-flexible model**.

4 Results

Table 1 and table 2 show our models performance for the development languages and also the test

	Development languages						
Language	Gold	Baseline		Non-flexible model		Flexible model	
	no. of slots	no. of slots	macro	no. of slots	macro	no. of slots	macro
Maltese	32	17	0.2029	2	0.013	254	0.0022
Persian	136	31	0.0605	2	0.0074	11	0.0155
Portuguese	76	34	0.3964	70	0.1275	1104	0.0109
Russian	16	19	0.4132	10	0.0706	387	0.0035
Swedish	11	15	0.4167	17	0.2282	588	0.0093

Table 1: Macro average results and the number of predicted slots for the Baseline model together with our Non-flexible and Flexible models, tested on development languages.

	Test languages						
Language	Gold	Baseline		Non-flexible model		Flexible model	
	no. of slots	no. of slots	macro	no. of slots	macro	no. of slots	macro
Basque	1658	27	0.0006	2	0.0001	30	0.0002
Bulgarian	54	34	0.3169	13	0.0415	138	0.0299
English	5	4	0.662	7	0.1729	51	0.0353
Finnish	141	21	0.055	108	0.0208	1169	0.0039
German	20	9	0.29	40	0.0498	425	0.007
Kannada	57	172	0.1512	1	0.0169	44	0.0427
Navajo	30	3	0.0327	2	0.002	38	0.0013
Spanish	70	29	0.2367	40	0.1084	225	0.0352
Turkish	120	104	0.1553	502	0.0071	1772	0.0011

Table 2: Macro average results and the number of predicted slots for the Baseline model together with our Non-flexible and Flexible models, tested on test languages.

languages. Unfortunately, we could not surpass the baseline model in any of the languages. We can say that among the development language results, Portuguese and Swedish are the ones that are best captured by the Non-flexible model. Considering the test languages, Spanish and English are the ones that were best modeled by the Non-flexible model.

It also seems that while the flexible model might have a better recall, the obtained result is not good enough, and therefore, it still requires some filtering.

5 Discussion and Future Work

We have presented our approach for automatically discovering morphological paradigms, given a text and list of lemmas. As mentioned above, our results are behind the official baseline, and therefore, there is a wide range of possibilities for improvement. We discuss some of them below.

We assumed each inflected form to be decomposable into a stem and a suffix. This could be, for example, sufficient for English or Spanish, but not for languages such as German that follow a two splits pattern:

$$\mathit{STEM}+\mathit{SUFFIX} \rightarrow \mathit{PREFIX}+\mathit{STEM}+\mathit{SUFFIX}'$$

In German, for example, participles are formed by prefixing *ge*:

$$\begin{array}{l|l} \mathit{play} \rightarrow \mathit{played} & \mathit{play}+\epsilon \rightarrow \mathit{play}+\mathit{ed} \\ \mathit{spielen} \rightarrow \mathit{gespielt} & \mathit{spiel}+\mathit{en} \rightarrow \mathit{ge}+\mathit{spiel}+t \end{array}$$

Apart from that, a much more straightforward estimate of the morphological richness r_m could, for example, be obtained by just considering the triple $l^1 = \hat{r}^1+s,\ l^2 = \hat{r}^2+s,\ l^3 = \hat{r}^3+s$ of optimally splitted distinct lemmas with the maximum number of common suffixes. Because these lemmas are most likely to be frequently used lemmas with regular inflection, the size of the union of their inflections would presumably yield a good estimate of r_m. Clustering of these triples could also help in identifying verb classes with distinct but regular inflection.

Moreover, splitting of compound verbs of the form X+V, with X typically a noun or verb, would certainly improve performance because the inflections of the verb V could be used for the typically less frequent compound verb X+V.

With respect to the writing system, the Basque bible follows old orthographical rules. On the other hand, the lemmas were written following more recent orthography rules. This lack of consistency makes the task a challenge, and we expect it to happen in other languages as well. This issue requires special attention, by maybe applying some preprocessing to the lemmas to accommodate to the old writing system (Etxeberria et al., 2019).

Also, we mentioned at the beginning of the article that we have not used any neural network based component, and these would be very useful for learning the morphophonological changes that commonly happen when inflecting words. Therefore, we would like to incorporate a Sequence-to-Sequence model at the end of our pipeline (Sutskever et al., 2014; Bahdanau et al., 2015; Vaswani et al., 2017).

Acknowledgments

We acknowledge the anonymous reviewers for their relevant comments and possible extra future directions, and also the organizers of the CoNLL–SIGMORPHON 2020 Shared Task for giving us the opportunity of participating in such a challenging task.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite-state Morphology: Xerox Tools and Techniques*. CSLI, Stanford University.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sabrina J. 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*, pages 1–27, Brussels. 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. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver. 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 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Alexander Erdmann, Micha Elsner, Shijie Wu, Ryan Cotterell, and Nizar Habash. 2020. The Paradigm Discovery Problem. *arXiv preprint arXiv:2005.01630*.

Izaskun Etxeberria, Iñaki Alegria, and Larraitz Uria. 2019. Weighted finite-state transducers for normalization of historical texts. *Natural Language Engineering*, 25(2):307–321.

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*, pages 634–643, San Diego, California. Association for Computational Linguistics.

John A Goldsmith, Jackson L Lee, and Aris Xanthos. 2017. Computational learning of morphology. *Annual Review of Linguistics*, 3:85–106.

Huiming Jin, Liwei Cai, Yihui Peng, Chen Xia, Arya D. McCarthy, and Katharina Kann. 2020. Unsupervised morphological paradigm completion. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Katharina Kann, Arya D. McCarthy, Garrett Nicolai, and Mans Hulden. 2020. The SIGMORPHON 2020 shared task on unsupervised morphological paradigm completion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2017. The LMU system for the CoNLL-SIGMORPHON 2017 Shared Task on Universal Morphological Reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 40–48.

Ronald M. Kaplan and Martin Kay. 1994. Regular Models of Phonological Rule Systems. *Computational Linguistics*, 20(3):331–378.

Kimmo Koskenniemi. 1983. Two-level morphology: A general computational model for word-form production and generation. *Publications of the Department of General Linguistics, University of Helsinki*.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sabrina J. Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019. The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244, Florence, Italy. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.