Morphological reinflection with weighted finite-state transducers

Alice Kwak and **Michael Hammond** and **Cheyenne Wing**
Dept. of Linguistics
U. of Arizona
Tucson, AZ 85721, USA
{alicekwak,hammond,cheyennewing}@arizona.edu

Abstract

This paper describes the submission by the University of Arizona to the SIGMORPHON 2023 Shared Task on typologically diverse morphological (re-)infection. In our submission, we investigate the role of frequency, length, and weighted transducers in addressing the challenge of morphological reinflection. We start with the non-neural baseline provided for the task and show how some improvement can be gained by integrating length and frequency in prefix selection. We also investigate using weighted finite-state transducers, jump-started from edit distance and directly augmented with frequency. Our specific technique is promising and quite simple, but we see only modest improvements for some languages here.

1 Introduction

This paper describes the submission by the University of Arizona to the SIGMORPHON 2023 Shared Task on typologically diverse morphological (re-)infection. The goal of the Shared Task is to model inflectional morphology. The specific task is to learn how to inflect for a language generally from a limited number of examples.

In this task, we are given $10,000$ examples of inflected forms in 27 languages along with the morphological category and the generally accepted lemma form. For example, in English, we have data as in Table 1.

...		
argue	V;NFIN	argue
argue	V;PRS;NOM(3,SG)	argues
argue	V;PST	argued
argue	V;V.PTCP;PRS	arguing
argue	V;V.PTCP;PST	argued
ascertain	V;NFIN	ascertain
ascertain	V;PRS;NOM(3,SG)	ascertains
ascertain	V;PST	ascertained
ascertain	V;V.PTCP;PRS	ascertaining
...		

Table 1: Some English training data

Morphosyntactic information is in Unimorph format (Guriel et al., 2022). The logic is that we are given complete paradigms for n lemmas for each language where the number of lemmas we see in the training data is a function of the size of the paradigms. Specifically, if paradigms are small, we see more lemmas than if paradigms are big.

The goal is to build a system that learns the relationship between lemmas L, morphosyntactic descriptions M, and inflected words W. The system effectively computes a function from $L \times M$ to W. More details on the task are given in Goldman et al. (2023).

The organizers provided two baseline systems, a neural and a non-neural one. We decided to focus our efforts on a non-neural solution and so we began our work by attempting to understand the non-neural baseline more clearly.

For comparison purposes, we ultimately submitted four sets of results: i) our implementation of the non-neural baseline; ii) a version of the non-neural baseline with adjustments for prefix frequency and length; iii) an approach using weighted finite-state transducers; and iv) an ensemble approach using both prefix frequency/length and weighted transducers.

In the following sections, we first review the structure of the baseline non-neural system. We then outline our approaches and present our results. We conclude with a discussion of shortcomings and next steps.[1]

2 Non-neural baseline

The non-neural baseline system (Cotterell et al., 2017) was inspired by (Liu and Mao, 2016). It

[1]All of our code is available at https://github.com/hammondm/sigmorphon23/.

Prefix	Stem	Suffix
∅	happ	y
un	happ	iness

Table 2: Alignment of *happy* and *unhappiness*

Prefixes	Suffixes
(<ha, <unha)	(py>, piness>)
(<, <un)	(>, ess>)
(<hap, <unhap)	(happy>, happiness>)
(<happ, <unhapp)	(>, s>)
(<h, <unh)	(>, ness>)
	(appy>, appiness>)
	(y>, iness>)
	(ppy>, ppiness>)
	(>, ss>)
	(>, >)

Table 3: Hypothesized rules for *happy* and *unhappiness* (angled brackets are word boundaries)

aligns lemmas and surface forms utilizing Hamming distance and Levenshtein distance and uses this alignment to hypothesize potential prefixes and suffixes.

For example, if the system were presented *happy* and *unhappiness*, it would hypothesize the morphological analysis in Table 2.[2]

This alignment would be used to extract potential prefix rules and suffix rules as in Table 3. During inference, the best prefix rule and suffix rule are chosen based on length and frequency. Specifically, the longest rule that produces the most identical forms is chosen.

Our first submission is essentially as described above.

3 Non-neural Baseline improvements

For our second submission, we made two revisions to the non-neural baseline system.

First, we replaced the input for extracting prefix rules, which was originally specified as the concatenation of lemma's prefix and surface form's root, with the concatenation of lemma's prefix and lemma's root. Given the alignment algorithm described above, this shouldn't have an effect in most cases, but it actually produced a small improvement. Presumably, this is because of cases where the lemma and word form do not share an obvious root.

Second, we changed the criteria for choosing the best prefix rule. The criteria for choosing the best rule had been set up asymmetrically for prefix rules and suffix rules. For suffix rules, the longest-matching rule(s) given an input and the morphosyntactic description was chosen as the best rule. If there are ties, the most frequent rule was chosen. For prefix rules, frequency had been the only criterion and the length of the match had not been considered. We revised the system so that the same criteria apply to both prefix rules and the suffix rules.

We were able to get a modest improvement in performance as a result of these revisions (non-neural baseline: 69.60%, revised system: 71.71%).[3]

4 Weighted finite-state transducers

We also built a system, inspired by the non-neural baseline described above, but which uses weighted finite-state transducers instead. Similar techniques have been tried before, for example, Durrett and DeNero (2013) and Forsberg and Hulden (2016). In fact, a number of them showed up in the 2016 version of this task, e.g. Alegria and Etxeberria (2016), Nicolai et al. (2016), Liu and Mao (2016), and King (2016).[4]

Durrett and DeNero (2013) learns a set of transformations: separate ones for prefixes, stems, and suffixes. They use a conditional random field (CRF) to combine and apply them.

Forsberg and Hulden (2016) generate probabilistic and non-probabilistic morphological analyzers in an automatic way by converting morphological inflection tables into unweighted and weighted FSTs.

Alegria and Etxeberria (2016) uses *Phonetisaurus*, a WFST-based system (Novak et al., 2012). This system directly learns a single WFST to model the lemma-to-word relation. The model thus includes a role for frequency, but not length. Morphosyntactic information is directly encoded in the WFST.

Nicolai et al. (2016) uses DirecTL+ (Jiampojamarn et al., 2008), a discriminative transducer that

[2]The example in the text is an instance of derivational morphology. Unfortunately, English does not have inflectional prefixes, so we use this.

[3]Our implementation of the non-neural baseline gets a slightly different macro average, so we cite the organizers' macro average here.

[4]Merzhevich et al. (2022) use transducers as well, but they are constructed by hand, not learned from data.

searches for a sequence of character transformation rules. It uses a version of the MIRA algorithm (McDonald et al., 2005) to assign weights to each feature. Transformations are N-gram-based and combined to produce surface forms.

Liu and Mao (2016) use a linear-chain conditional random field model with contextual features, e.g. what is a consonant or vowel.

King (2016) uses conditional random fields as well. Separate edit rules are induced from edit distance comparisons and combined at inference. Other features like position in the string were also incorporated.

In our model, we use edit distance to calculate the precise overlap between a lemma and a surface form and then build a weighted finite-state transducer (WFST) from that that specifies changes with interleaving variables. The weights penalize degrees of mismatch with the variables.

For example, take a lemma-word pair like *break* and *broken*. First, we use edit distance to calculate an optimal alignment. We then replace all identical spans with variables that penalize mismatches. In this example, *br* would be a variable and *k* would be a variable. Our transducers are implemented in *pyfoma*[5]. Using the formalism of that system, the resulting transducer would be specified as below:

$$(.*\mathrm{<n>}|\mathrm{br})(\mathrm{ea}:\mathrm{o})(.*\mathrm{<n>}|\mathrm{k})('':\mathrm{en})\mathrm{<m>}$$

Here there is a penalty associated with not matching the spans where the two forms align. In the formalism above, we've specified these as n to indicate that we experimented with different weighting options. There is a penalty associated with the rule as a whole, indicated above as m. This was used to incorporate different possible costs for the length of the rule. Again, we tried different options here, but the general strategy was to penalize shorter less frequent rules.

In training, we built transducers in this way for all training items, separated by morphosyntactic descriptions. For inference, we generated all possible outputs for a lemma with the WFSTs for that morphosyntactic description and chose the one that had the lowest cost.

We expected such a system would be better able to capture *nonconcatenative* morphological systems, systems where morphological categories might be marked by stem-internal changes as opposed to prefixes or suffixes. In fact, as we discuss below, this was not the case.

[5]https://github.com/mhulden/pyfoma

Based on development split performance, we saw that the frequency of forms played a role: as with the baseline system, more frequent output forms were preferred. To handle this, we adjusted our weighting scheme so that if multiple WFSTs produced the same output, those got lowered scores.

Our approach differs from previous WFST-based approaches in three main respects. First, our alignment and overall system is extremely simple, as described above. Second, our weights are naive, not trained, and assigned on the basis of a general theory of what should matter, as described above. Finally, our system employs only transducers.

One strength of our system is that it's very straightforward and easy to manipulate the weights. It can be used to test the effect each factor (e.g., form frequency, length of rules, etc.) has over the system's performance.

5 Ensemble system

We found that our system generally did not perform as well as the non-neural baseline or our revision of it, but we saw improved performance for some languages with development data, specifically Japanese, Armenian, and Italian. Therefore we also submitted a ensemble system. where we generated test outputs using our improved baseline and our WFST system and selected the test results based on development data performance.

6 Results

Results for our four submissions are given in Table 4.

Submission 1 is our execution of the non-neural baseline. It is here simply for comparison purposes.[6]

Submission 2 is our simple adaptation of the non-neural baseline to make prefixation and suffixation sensitive to the same variables of length and frequency of surface forms. The adapted system performed better than the non-neural baseline (67.1% vs. 71.7%).

[6]Just for completeness, our version of the non-neural baseline differs from the organizer's in one key line where lists of strings are zipped together. In the organizer's version that object is then converted directly to a list; in our version, it is not.

Submission 3 is the WFST-based system. It does not perform very well in general (56.1%), but, as noted above, it does better than the systems 1 and 2 in a couple of cases: Armenian, Italian, and Japanese.

Finally, submission 4 is the ensemble system, where we draw on systems 2 and 3 depending on which performed better with development data.

7 Discussion

We focused our efforts on a non-neural approach and thus did not expect competitive results. That said, we did manage to improve over the non-neural baseline. Our intention was to understand more deeply how morphological systems could be modeled in the simplest finite-state terms. To this end, we conducted several experiments with our WFST system.

One of the experiments we did is to create the WFST with individual characters, instead of spans, as variables. In our submitted system, spans that are identical in a lemma-word pair are replaced with variables. We revised the system to replace individual matching characters with variables. For example, take the lemma-word pair *break* and *broken* once again. In our submitted system, identical spans *br* and *k* are replaced with variables that penalize mismatches. In the revised system, individual characters *b*, *r*, and *k* are replaced with variables. Thus the first variable in the formula below is replaced as shown.

$$(.*\text{<n>}|\text{br})(\text{ea}:\text{o})(.*\text{<n>}|\text{k})('':\text{en})\text{<m>}$$

not split:	$(.*\text{<n>}\|\text{br})$
split:	$(.*\text{<n>}\|\text{b})(.*\text{<n>}\|\text{r})$

The motivation for this experiment was to see if penalizing each unmatched character, rather than the whole span, would enhance the system's performance. Our hypothesis was that penalizing individual characters would improve the system, as it would give a more specific penalty to an unmatched span.

This was not the case. At least for the various weightings we tried, the individual character variable versions did not perform as well as system 3 above.

In addition, the individual variable versions entailed much larger transducers and much longer run times. For the systems we submitted, running the languages in parallel meant a complete run always took less than an hour. For the individual variable versions, running the languages in parallel took over 15 hours on our campus supercomputer.

Our expectation is that with a compiled transducer system like Foma or OpenFST and with more aggressive parallelization, we could reduce this runtime significantly.

Another experiment we did is to adjust weights based on the frequency of the form produced by the candidate WFSTs. During error analysis, we found out that there are many WFST candidates producing the same form. In many cases these frequent forms were the correct ones, but they were not selected as the optimal forms due to high weights. In order to mitigate this issue, we tried adjusting the weights of the WFSTs based on the frequency of the form the transducer produced. As a result of this adjustment, we were able to obtain a boost of approximately 5% in system performance on development data (from 51% to 56%).

8 Conclusion

In conclusion, we developed three non-neural models. The first combined frequency and length in the selection of prefixes. The second used WFSTs built from edit distance alignments. The third model combined the results of the first two models.

The direct baseline changes resulted in overall improvements, but the WFST system did not. However, there were specific language improvements from the WFST solution and we were able to incorporate these in our ensemble system.

9 Limitations

While the WFST model didn't perform very well overall, our sense is that it is worth pursuing further. Specifically, there are several moves worth exploring.

First, we should move to a compiled system so that we can test the "individual variable" models more thoroughly.

Second, we should try models where we set the variable weights by training, rather than naively in advance.

Third, in an individual variable setting, it would be promising to weight the variables by locality. Specifically, do mismatched variables have more effect when they are closer to where the changes happen? Similarly, we might adjust the granularity of the variables as a function of position, with

Language	1	2	3	4
Arabic, Gulf	0.308	0.527	0.345	0.527
Amharic	0.654	0.74	0.599	0.74
Arabic, Egyptian	0.772	0.808	0.757	0.808
Belarusian	0.681	0.645	0.462	0.645
Danish	0.895	0.874	0.648	0.874
German	0.798	0.779	0.599	0.779
English	0.966	0.962	0.67	0.962
Finnish	0.808	0.806	0.482	0.806
French	0.777	0.763	0.767	0.763
Ancient Greek	0.526	0.548	0.404	0.548
Hebrew	0.309	0.653	0.347	0.653
Hebrew (Unvoc)	0.645	0.767	0.516	0.767
Hungarian	0.747	0.747	0.459	0.747
Armenian	0.863	0.862	0.889	0.889
Italian	0.75	0.636	0.78	0.78
Japanese	0.641	0.641	0.67	0.67
Georgian	0.82	0.821	0.717	0.821
Khaling	0.545	0.531	0.278	0.531
Macedonian	0.916	0.908	0.649	0.908
Navajo	0.358	0.418	0.237	0.418
Russian	0.86	0.856	0.668	0.856
Sanskrit	0.622	0.621	0.47	0.621
Sami	0.56	0.497	0.301	0.497
Spanish	0.878	0.874	0.863	0.874
Albanian	0.193	0.781	0.738	0.781
Swahili	0.605	0.65	0.562	0.65
Turkish	0.646	0.646	0.281	0.646
macro	0.671	0.717	0.561	0.724

Table 4: Language-by-language results for our four submissions

single-character variables sometimes and variable spans in other cases.

Fourth, We had individual WFSTs for each lemma, but with a compiled system it makes sense to put them all together into a single WFST.

References

Iñaki Alegria and Izaskun Etxeberria. 2016. EHU at the SIGMORPHON 2016 shared task. a simple proposal: Grap heme-to-phoneme for inflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 27–30, Berlin, Germany. 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.

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*, pages 1185–1195, Atlanta, Georgia. Association for Computational Linguistics.

Markus Forsberg and Mans Hulden. 2016. Learning transducer models for morphological analysis from example inflections. In *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*, pages 42–50, Berlin, Germany. Association for Computational Linguistics.

Omer Goldman, Khuyagbaatar Batsuren, Khalifa Salam, Aryaman Arora, Garrett Nicolai, Reyt Tsarfaty, and Ekaterina Vylomova. 2023. SIGMORPHON–UniMorph 2023 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 20th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, Toronto, Canada. Association for Computational Linguistics.

David Guriel, Omer Goldman, and Reut Tsarfaty. 2022. Morphological reinflection with multiple arguments: An extended ann otation schema and a Georgian case study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 196–202, Dublin, Ireland. Association for Computational Linguistics.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proceedings of ACL-08: HLT*, pages 905–913, Columbus, Ohio. Association for Computational Linguistics.

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*, pages 49–53, Berlin, Germany. Association for Computational Linguistics.

Ling Liu and Lingshuang Jack Mao. 2016. Morphological reinflection with conditional random fields and unsup ervised features. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 36–40, Berlin, Germany. Association for Computational Linguistics.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 91–98, Ann Arbor, Michigan. Association for Computational Linguistics.

Tatiana Merzhevich, Nkonye Gbadegoye, Leander Girrbach, Jingwen Li, and Ryan Soh-Eun Shim. 2022. SIGMORPHON 2022 task 0 submission description: Modelling morpholo gical inflection with data-driven and rule-based approaches. In *Proceedings of the 19th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 204–211, Seattle, Washington. Association for Computational Linguistics.

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*, pages 31–35, Berlin, Germany. Association for Computational Linguistics.

Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. In *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, pages 45–49, Donostia–San Sebastián. Association for Computational Linguistics.