Ensemble Self-Training for Low-Resource Languages: Grapheme-to-Phoneme Conversion and Morphological Inflection

Xiang Yu, Ngoc Thang Vu, Jonas Kuhn
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart, Germany
firstname.lastname@ims.uni-stuttgart.de

Abstract

We present an iterative data augmentation framework, which trains and searches for an optimal ensemble and simultaneously annotates new training data in a self-training style. We apply this framework on two SIGMORPHON 2020 shared tasks: grapheme-to-phoneme conversion and morphological inflection. With very simple base models in the ensemble, we rank the first and the fourth in these two tasks. We show in the analysis that our system works especially well on low-resource languages. The system is available at https://www.ims.uni-stuttgart.de/en/institute/team/Yu-00010/.

1 Introduction

The vast majority of languages in the world have very few annotated dataset available for training natural language processing models, if at all. Dealing with the low-resource languages has sparked much interest in the NLP community (Garrette et al., 2013; Agić et al., 2016; Zoph et al., 2016).

When annotation is difficult to obtain, data augmentation is a common practice to increase training data size with reasonable quality to feed to powerful models (Ragni et al., 2014; Bergmanis et al., 2017; Silfverberg et al., 2017). For example, the data hallucination method by Anastasopoulos and Neubig (2019) automatically creates non-existing "words" to augment morphological inflection data, which alleviates the label bias problem in the generation model. However, the data created by such method can only help regularize the model, but cannot be viewed as valid words of a language.

Orthogonal to the data augmentation approach, another commonly used method to boost model performance without changing the architecture is ensembling, i.e., by training several models of the same kind and selecting the output by majority voting. It has been shown that a key to the success of ensembling is the diversity of the base models (Surdeanu and Manning, 2010), since models with different inductive biases are less likely to make the same mistake.

In this work, we pursue a combination of both directions, by developing a framework to search for the optimal ensemble and simultaneously annotate unlabeled data. The proposed method is an iterative process, which uses an ensemble of heterogeneous models to select and annotate unlabeled data based on the agreement of the ensemble, and use the annotated data to train new models, which are in turn potential members of the new ensemble. The ensemble is a subset of all trained models that maximizes the accuracy on the development set, and we use a genetic algorithm to find such combination of models.

This approach can be viewed as a type of self-training (Yarowsky, 1995; Clark et al., 2003), but instead of using the confidence of one model, we use the agreement of many models to annotate new data. The key difference is that the model diversity in the ensemble can alleviate the confirmation bias of typical self-training approaches.

We apply the framework on two of the SIGMORPHON 2020 Shared Tasks: grapheme-to-phoneme conversion (Gorman et al., 2020) and morphological inflection (Vylomova et al., 2020). Our system rank the first in the former and the fourth in the latter.

While analyzing the contribution of each component of our framework, we found that the data augmentation method does not significantly improve the results for languages with medium or large training data in the shared tasks, i.e., the advantage of our system mainly comes from the massive ensemble of a variety of base models. However, when we simulate the low-resource scenario or consider only the low-resource languages, the benefit of data augmentation becomes prominent.

2 Ensemble Self-Training Framework

2.1 General Workflow

In this section we describe the details of our framework. It is largely agnostic to the type of supervised learning task, while in this work we apply it on two sequence generation tasks: morphological inflection and grapheme-to-phoneme conversion. The required component includes one or more types of base models and large amount of unlabeled data. Ideally, the base models should be simple and fast to train with reasonable performance, and as diverse as possible, i.e., models with different architectures are better than the same architecture with different random seeds.

The workflow is described in Algorithm 1. Initially, we have the original training data L_0, unlabeled data U, and several base model types $T^{1...k}$. In each iteration n, there are two major steps: (1) ensemble training and (2) data augmentation. In the ensemble training step, we train each base model type on the current training data L_n to obtain the models $m_n^{1...k}$, and add them into the model pool (line 4-8). We then search for an optimal subset of the models from the pool as the current ensemble, based on its performance on the development set (line 9). In the data augmentation step, we sample a batch of unlabeled data (line 10), then use the ensemble to predict and select a subset of the instances based on the agreement among the models (line 11). The selected data are then aggregated into the training set for later iterations (line 12-13).

Algorithm 1 Ensemble Self-Training (EST)

```
 1: function EST($L, U, T$)
Require: labeled data $L$
Require: unlabeled data $U$
Require: tools $T$
 2:     Initial data $L_0 = L$
 3:     Model pool $M = \emptyset$
 4:     for $n : 0...N$ do
 5:         for $t^k \in T$ do
 6:             $m_n^k = \textsc{Train}(t^k, L_n)$
 7:             $M = M \cup \{m_n^k\}$
 8:         end for
 9:         $E = \textsc{SearchEnsemble}(M)$
10:         Sample $u \sim U$
11:         $l = \textsc{SelectData}(E, u)$
12:         $L_{n+1} = \textsc{AggregateData}(L_n, l)$
13:         $U = U - l$
14:     end for
15:     return $E$, $L_k$
16: end function
```

2.2 Ensemble Search

Simply using all the models as the ensemble would be not only slow but also inaccurate, since too many inferior models might even mislead the ensemble, therefore searching for the optimal combination is needed. However, an exact search is not feasible, since the number of combinations grows exponentially. We use the genetic algorithm for heterogeneous ensemble search largely following Haque et al. (2016). In the preliminary experiments, the genetic algorithm consistently finds better ensembles than random sampling or using all models.

We use a binary encoding such as 0100101011 to represent an ensemble combination (denoted as an individual in genetic algorithms), where each bit encodes whether to use one particular model.

As we aim to maximizing the prediction accuracy of the ensemble, we define the fitness score of an individual as the accuracy on the development set by the ensemble represented by the individual.

Initially, we generate 100 random individuals into a pool, which is maintained at the size of 100. Whenever a new individual enters the pool, the individual with the lowest fitness score will be removed.

Each new individual is created through three steps: parent selection, crossover, and mutation. Both parents are selected in a tournament style, in which we sample 10 individuals from the pool, and take the one with the highest fitness score. In the crossover process, we take each bit randomly from one parent with a rate of 60%, and 40% from the other. In the mutation process, we flip each bit of the child with a probability of 1%. To ensure the efficiency of the ensemble, we also limit the number of models in the combination to 20: if a newly evolved combination exceeds 20 models, we randomly reduce the number to 20 before evaluating the fitness.

In each search, we evolve 100,000 individuals, and return the one with the highest fitness score. Since the data size is relatively small, the ensemble search procedure typically only takes a few seconds.

2.3 Data Selection and Aggregation

In each iteration, we use the current optimal ensemble to predict a batch of new data, and select a subset as additional data to train models in the next iteration.

There are various heuristics to select new data, with two major principles to consider: (1) one should prefer the instances with higher agreement among the models, since they are more likely to be correct; (2) instances with unanimous agreement might be too trivial and does not provide much new information to train the models.

To strike a balance between the two considerations, we first rank the data by the agreement, but only take at most half of the instances with unanimous agreement as new annotated data. Concretely, we sample 20,000 instances to predict, and use at most 3,600 instances as new data if their predictions have over 80% agreement, among which, at most 1,800 instances have 100% agreement. Note that we chose the data size of 3,600 because it is the training data size in the grapheme-to-phoneme conversion task, and we used the same setting for the morphological inflection task without tuning.

There are also different ways to aggregate the new data. One could simply accumulate all the selected data, resulting in much larger training data in the later iterations, which might slow down the training process and dilute the original data too much. Alternatively, one could append only the selected data from the current iteration to the original data, which might limit the potential of the models.

Again, we took the middle path, in which we keep half of all additional data from the previous iteration together with the selected data in the current iteration. For example, there are 3600 additional instances produced in iteration 0, $3600/2 + 3600 = 5400$ in iteration 1, $5400/2 + 3600 = 6300$ in iteration 2, and the size eventually converges to $3600 \times 2 = 7200$.

3 Grapheme-to-Phoneme Conversion

3.1 Task and Data

We first apply our framework on the grapheme-to-phoneme conversion task (Gorman et al., 2020), which includes 15 languages from the WikiPron project (Lee et al., 2020) with a diverse typological spectrum: Armenian (arm), Bulgarian (bul), French (fre), Georgian (geo), Hindi (hin), Hungarian (hun), Icelandic (ice), Korean (kor), Lithuanian (lit), Modern Greek (gre), Adyghe (ady), Dutch (dut), Japanese hiragana (jpn), Romanian (rum), and Vietnamese (vie).

As preprocessing, we romanize the scripts of Japanese and Korean,[1][2] which show improvements in preliminary experiments. The reason is that the Japanese Hiragana and Korean Hangul characters are both syllabic, in which one grapheme typically corresponds to multiple phonemes, and by romanizing them (1) the alphabet size is reduced, and (2) the length ratio of the source and target sequences are much closer to 1:1, which empirically improve the quality of the alignment.

As unlabeled data, we use word frequency lists,[3] which are mostly extracted from OpenSubtitles (Lison and Tiedemann, 2016). For the two languages we did not find in OpenSubtitles, Adyghe is obtained from the corpus by Arkhangelskiy and Lander (2016),[4] and Georgian is obtained from several text corpora.[5][6]

Since the word lists are automatically extracted from various sources with different methods and quality, we filter them by the alphabet of the training set of each language, and keep at most 100,000 most frequent words.

3.2 Models

As the framework desires the models to be as diverse as possible to maximize its benefit, we employ four different types of base models with different inductive biases.

The first type is the Finite-State-Transducer (FST) baseline by Lee et al. (2020), based on the pair n-gram model (Novak et al., 2016).

The other three types are all variants of Seq2Seq models, where we use the same BiLSTM encoder to encode the input grapheme sequence. The first one is a vanilla Seq2Seq model with attention (`attn`), similar to Luong et al. (2015), where the decoder applies attention on the encoded input and use the attended input vector to predict the output phonemes.

The second one is a hard monotonic attention model (`mono`), similar to Aharoni and Goldberg (2017), where the decoder uses a pointer to select the input vector to make a prediction: either produc-

[1] https://pypi.org/project/pykakasi/
[2] https://pypi.org/project/hangul-romanize/
[3] https://github.com/hermitdave/FrequencyWords/
[4] https://github.com/timarkh/uniparser-grammar-adyghe
[5] https://github.com/akalongman/geo-words
[6] Georgian is actually in OpenSubtitles, but we accidentally missed it because of a confusion with the language code.

ing a phoneme, or moving the pointer to the next position. The monotonic alignment of the input and output is obtained with the Chinese Restaurant Process following Sudoh et al. (2013), which is provided in the baseline model of the SIGMORPHON 2016 Shared Task (Cotterell et al., 2016).

The third one is essentially a hybrid of hard monotonic attention model and tagging model (`tag`), i.e., for each grapheme we predict a short sequence of phonemes that is aligned to it. It relies on the same monotonic alignment for training. This model is different from the previous one in that it can potentially alleviate the error propagation problem, since the short sequences are non-autoregressive and independent of each other, much like tagging.

For each of the three models, we further create a reversed variant, where we reverse the input sequence and subsequently the output sequence. On average, the best model types are the tagging models of both directions.

Since we need to train many base models, we keep their sizes at a minimal level: the LSTM encoder and decoder both have one layer, all dimensions are 128, and no beam search is used. As a result, each base model has about 0.3M parameters and takes less than 10 minutes to train on a single CPU core.

3.3 Experiments

With the ensemble self-training framework, we train 14 base models at each iteration: FST models with 3-grams and 7-grams (`fst-3`, `fst-7`), two instances for each direction of the attention model (`attn-l2r`, `attn-r2l`), hard monotonic model (`mono-l2r`, `mono-r2l`), and tagging model (`tag-l2r`, `tag-r2l`).

Table 1 shows the number of iterations when the optimal ensemble is found and the number of models it contains, as well as the Word Error Rate (WER) and Phone Error Rate (PER) on the test set, in comparison to the Seq2Seq baseline provided by the organizer. Generally, our system outperforms the strong baseline in 13 out of 15 languages, and the gap for Korean is especially large, due to the romanization in our preprocessing. For three languages (Hungarian, Japanese, and Lithuanian), the best ensemble is in the 0-th iteration, which means the augmented data for them is not helpful at all.

Our ensemble system rank the first in terms of both WER and PER on the test set, with an average WER of 13.8 and PER of 2.76. However, a large ensemble of simple models is not exactly comparable with other single-model systems, and it is thus difficult to derive a conclusion from the evaluation alone. We are more interested in understanding how much of the improvement comes from the ensemble and its model diversity and how much from the data augmentation process.

			IMS		Seq2Seq	
	#iter	#model	WER	PER	WER	PER
ady	4	20	**25.33**	**5.79**	28.00	6.53
arm	5	20	**12.67**	**2.94**	14.67	3.49
bul	4	10	**22.22**	**4.85**	31.11	5.94
dut	2	13	**13.56**	**2.36**	16.44	2.94
fre	1	17	6.89	1.60	**6.22**	**1.32**
geo	6	20	**24.89**	**4.57**	26.44	5.14
gre	1	12	**18.67**	**2.97**	18.89	3.30
hin	1	20	**5.11**	**1.20**	6.67	1.47
hun	0	5	**5.11**	**1.12**	5.33	1.18
ice	5	20	**9.33**	**2.04**	10.00	2.36
jpn	0	6	**5.33**	**1.26**	7.56	1.79
kor	4	8	**26.22**	**4.38**	46.89	16.78
lit	0	5	20.00	3.63	**19.11**	**3.55**
rum	1	8	**10.22**	**2.23**	10.67	2.53
vie	5	20	**1.56**	**0.48**	4.67	1.52
AVG	3	14	13.81	2.76	16.84	3.99

Table 1: Evaluation on the test set of the grapheme-to-phoneme conversion task, comparing our system with the best performing seq2seq baseline. The first two columns are the number of iterations when the best ensemble is found and the number of base models in the ensemble.

For this purpose, we run our framework in two additional scenarios. In the first scenario, we reduce the diversity of the models (denoted as -diversity), where we only use the base model `tag-l2r` and `tag-r2l`, which performs the best among others, but keep the same number of models trained in each iteration as before. In the second scenario, we do not perform data augmentation (denoted as -augmentation), i.e., all models are trained on the same original training data in each iteration.

Table 2 shows the WER on the development set of the default scenario and the two experimental scenarios. For each scenario, we show the average WER of all models and the WER of the ensemble from the initial iteration and the best iteration.

We can observe three trends in the table. (1) In all scenarios, there is a large gap between the aver-

	default				-diversity				-augment			
	average		ensemble		average		ensemble		average		ensemble	
	init	best	init	best	init	best	init	best	init	best	init	best
ady	28.9	27.7	22.4	21.6	26.6	27.2	22.9	22.2	28.7	28.1	22.7	20.9
arm	18.8	17.4	13.1	11.3	16.1	15.4	12.2	11.8	18.7	18.1	12.2	10.7
bul	36.8	36.2	25.3	20.0	35.5	35.5	27.6	23.8	37.3	36.1	24.2	18.7
dut	19.5	18.8	11.8	10.4	18.5	18.8	12.2	10.9	19.7	19.6	11.6	9.8
fre	15.1	15.7	6.0	5.6	13.2	13.6	6.7	6.2	15.6	15.2	7.1	5.1
geo	26.9	26.7	20.2	17.8	26.6	25.1	20.7	18.4	27.0	26.8	19.6	16.7
gre	20.1	18.4	13.8	12.7	17.3	16.8	12.7	11.3	19.9	19.8	12.9	11.8
hin	9.7	9.0	4.0	3.6	8.1	6.9	4.2	4.0	9.7	9.3	4.0	3.6
hun	4.5	4.5	2.0	2.0	4.0	3.9	2.4	2.2	4.7	4.7	2.4	2.4
ice	15.3	14.1	6.4	5.6	11.9	11.4	5.6	5.3	14.8	14.6	6.2	5.6
jpn	8.0	8.0	6.0	6.0	7.7	7.7	6.2	6.2	8.0	8.0	5.8	5.8
kor	25.9	23.4	16.2	14.4	20.9	20.7	16.9	16.0	25.9	25.6	16.4	14.2
lit	24.5	24.5	18.4	18.4	22.7	22.7	18.2	18.2	24.4	24.9	18.2	16.7
rum	14.6	13.7	10.2	9.8	12.2	12.2	10.0	9.3	14.4	14.5	9.8	8.7
vie	6.0	5.8	1.1	0.9	5.3	5.3	2.0	2.0	6.0	6.2	1.3	0.7
AVG	18.3	17.6	11.8	10.7	16.4	16.2	12.0	11.2	18.3	18.1	11.6	10.1

Table 2: WER on the development set in the three scenarios (default, reduced diversity, and without data augmentation). In each scenario, we show the average model performance and the ensemble performance in the first iteration and the best iteration.

age model performance and the ensemble performance, which clearly demonstrates the benefit of the ensemble. (2) In the -diversity scenario, the average model performance is better than the default scenario, but the ensemble performance is worse than the default scenario, which demonstrates the importance of the model diversity. (3) The average model performance in the default scenario has clear improvement as opposed to the random fluctuation in the -augmentation scenario, which means that the data augmentation can indeed benefit some individual models. However, to our surprise and disappointment, the ensemble performance of the -augmentation scenario is even slightly better than the default scenario, which casts a shadow over the data augmentation method in this framework.

As our framework is designed for low-resource languages, and the data size of 3,600 in the task is already beyond low-resource, we therefore experiment in a simulated low-resource scenario.[7] For each language, we randomly sample 200 instances as the new training data, while ensuring that all graphemes and phonemes in the training data appear at least once.

[7]Consider the Swadesh list (Swadesh, 1950) with only 100-200 basic concepts/words, which could be thought of as a typical low-resource scenario. In the WikiPron collection, more than 20% of the 165 languages have less than 200 words.

Table 3 shows the WER of the default and -augment scenario in the low-resource experiment. Similar to the previous experiment, the ensemble greatly reduces errors of individual models. More importantly, the individual models benefit significantly from the augmented data (from 54.2 to 35.5), and the final ensemble further reduces the error rate to 25.2. The WER in the default scenario is much better than the -augment scenario (25.2 vs 29.2), which means that the data augmentation is indeed beneficial when the training data is scarce.

4 Morphological Inflection

4.1 Task and Data

We also apply our framework on the morphological inflection task (Vylomova et al., 2020), where the input is a combination of lemmata and morphological tags according to the UniMorph schema (Sylak-Glassman et al., 2015), and the output is the inflected word forms. There are 90 languages with various data sizes, ranging from around 100 to 100,000.

As unlabeled data for the augmentation process, we simply recombine the lemmata and morphological tags of the same category in the training set (i.e.,

	default				-augment			
	average		ensemble		average		ensemble	
	init	best	init	best	init	best	init	best
ady	62.3	41.3	44.4	30.0	63.0	62.1	44.4	37.8
arm	42.6	30.2	28.0	22.9	42.5	41.9	28.9	23.3
bul	68.8	58.0	53.6	48.4	67.4	66.8	53.3	47.3
dut	64.9	37.8	45.6	27.6	64.7	63.2	43.1	32.4
fre	62.0	34.0	34.9	18.9	61.8	61.3	35.1	29.3
geo	40.5	34.4	29.8	26.0	40.6	39.6	30.4	24.7
gre	56.8	37.7	37.3	28.0	57.4	55.8	39.3	31.3
hin	53.8	22.2	32.2	12.7	53.5	52.8	33.8	24.4
hun	42.2	19.7	21.8	12.7	42.8	41.6	21.6	16.7
ice	71.1	51.1	53.8	42.9	73.6	70.4	55.8	49.8
jpn	41.4	16.5	19.8	11.1	42.1	40.4	21.3	15.6
kor	53.4	39.4	36.9	30.4	54.6	53.1	38.2	32.9
lit	66.3	51.8	49.1	38.4	67.4	65.9	48.7	39.8
rum	37.5	24.7	22.9	16.4	38.1	37.1	23.1	18.2
vie	50.3	33.4	23.8	11.1	50.6	49.3	21.6	14.4
AVG	54.2	35.5	35.6	25.2	54.7	53.4	35.9	29.2

Table 3: WER on the development set for the simulated low-resource experiment in the scenarios with and without data augmentation. In each scenario, we show the average model performance and the ensemble performance in the first iteration and the best iteration.

a verb lemma only combines with all morphological tags for verbs), with a maximum size of 100,000 for each language. For many languages, however, the recombination is as scarce as the original data since they are from (almost) complete inflection paradigms of a few lemmata. In total, we obtained 1,422,617 instances, which is slightly smaller than the training set with 1,574,004 instances. Since the additional data come directly from the original training data, we consider it the restricted setting, where no external data sources or cross-lingual methods are used.

4.2 Models

Due to our late start in this task, we only implemented two types of base models, paired with left-to-right and right-to-left generation order. The first type is a Seq2Seq model with soft attention, very similar to the one in the grapheme-to-phoneme conversion task, except that an additional BiLSTM is used to encode the morphological tags. The second type is a hard monotonic attention model, also similar as before, but instead of using the alignment with the Chinese Restaurant Process, we use Levenshtein edit scripts to obtain the target sequence,

Model	Accuracy
CULing-01-0	0.912
deepspin-02-1	0.909
uiuc-01-0	0.905
IMS-00-0	0.892
mono	0.858
trm	0.901
mono-aug	0.888
trm-aug	0.903

Table 4: Evaluation on the test set of the morphological inflection task, comparing our system to three winning systems and four baselines.

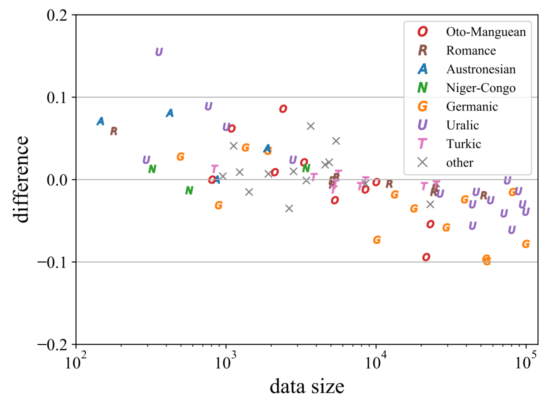
since the input and the output share the same alphabet. At each step, the model either outputs a character from the alphabet, or copies the currently pointed input character, or advances the input pointer to the next position. In total, we train 8 models per iteration, i.e., two models with different random seeds for each variant. The hyperparameters are largely the same as in the previous task, and each model has about 0.5M parameters.

4.3 Experiments

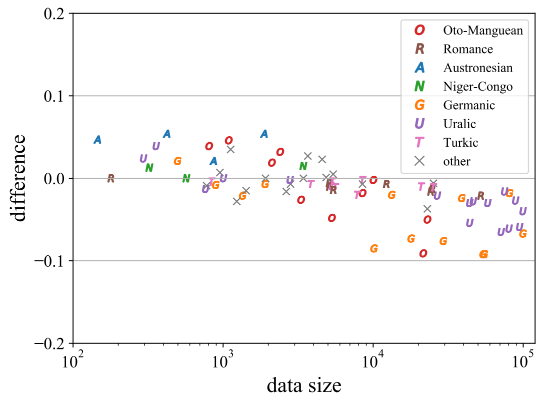
Table 4 compares the average test accuracy between our system (IMS-00-0) and the systems of the winning teams as well as the baselines. The baselines include a hard monotonic attention model with latent alignment (Wu and Cotterell, 2019) and a carefully tuned transformer (Vaswani et al., 2017; Wu et al., 2020), noted as `mono` and `trm`. They are additionally trained with augmented data by Anastasopoulos and Neubig (2019), noted as `mono-aug` and `trm-aug`.

On average, our system ranks the fourth among the participating teams and the third in the restricted setting (without external data source or cross-lingual methods). It outperforms the hard monotonic attention baseline, but not the transformer baseline. More details on the systems and their comparisons are described in Vylomova et al. (2020). Compared to the previous task, we used fewer base models, in terms of both number and diversity, which partly explains the relatively lower ranking.

In this task, the data size ranges across several magnitude for different languages. We thus analyze the performance difference of our system against the two baselines with their own data augmentation



(a) Hard Monotonic Attention



(b) Transformer

Figure 1: Performance difference between our system and the two baselines with data augmentation, with respect to the training data size.

(`mono-aug` and `trm-aug`) with respect to the original training data size, as illustrated in Figure 1. We removed the trivial cases in which both models achieved 100% accuracy.

Clearly, our system performs better for languages with smaller training data size, while losing to the powerful baseline models when the data size is large. This again demonstrates the benefit of our framework for low-resource languages.

We also mark the major language families to see whether they play a role in the performance difference, since different inductive biases might work differently on particular language families. For example, the right-to-left generation order might work better on languages with inflectional prefixes. However, we could not find any convincing patterns regarding language families in the plot, i.e., there is not a language family in the data set where our model always performs better or worse than the baseline. The only exception is the Austronesian family, where our system generally outperforms the baselines, but they all have relatively small data size, which is a more probable explanation.

Note that our augmentation method is theoretically orthogonal to the hallucination method (Anastasopoulos and Neubig, 2019), and could be combined to further improve the performance of the baseline models for low-resource languages.

5 Conclusion

We present an ensemble self-training framework and apply it on two sequence-to-sequence generation tasks: grapheme-to-phoneme conversion and morphological inflection. Our framework includes an improved self-training method by optimizing and utilizing the ensemble to obtain more reliable training data, which shows clear advantage on low-resource languages. The optimal ensemble search method with the genetic algorithm easily accommodates the inductive biases of different model architectures for different languages.

As a potential future direction, we could incorporate the framework into the scenario of active learning to reduce annotator workload, i.e., by suggesting plausible predictions to minimize the need of correction.

Acknowledgements

This work was in part supported by funding from the Ministry of Science, Research and the Arts of the State of Baden-Württemberg (MWK), within the CLARIN-D research project.

References

Željko Agić, Anders Johannsen, Barbara Plank, Héctor Martínez Alonso, Natalie Schluter, and Anders Søgaard. 2016. Multilingual projection for parsing truly low-resource languages. *Transactions of the Association for Computational Linguistics*, 4:301–312.

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)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Antonios Anastasopoulos and Graham Neubig. 2019. Pushing the limits of low-resource morphological inflection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 984–996, Hong Kong, China. Association for Computational Linguistics.

Timofey Arkhangelskiy and Yury Lander. 2016. Developing a Polysynthetic Language Corpus: Problems and Solutions. In *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialogue 2016"*, pages 40–49.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 31–39.

Stephen Clark, James Curran, and Miles Osborne. 2003. Bootstrapping POS-taggers using unlabelled data. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 49–55.

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.

Dan Garrette, Jason Mielens, and Jason Baldridge. 2013. Real-world semi-supervised learning of pos-taggers for low-resource languages. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 583–592.

Kyle Gorman, Lucas F.E. Ashby, Aaron Goyzueta, Shijie Wu, and Daniel You. 2020. The SIGMORPHON 2020 shared task on multilingual grapheme-to-phoneme conversion. In *SIGMORPHON*.

Mohammad Nazmul Haque, Nasimul Noman, Regina Berretta, and Pablo Moscato. 2016. Heterogeneous Ensemble Combination Search Using Genetic Algorithm for Class Imbalanced Data Classification. *PloS one*, 11(1):e0146116.

Jackson L. Lee, Lucas F.E. Ashby, M. Elizabeth Garza, Yeonju Lee-Sikka, Sean Miller, Alan Wong, Arya D. McCarthy, and Kyle Gorman. 2020. Massively multilingual pronunciation mining with WikiPron. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4216–4221, Marseille.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting Large Parallel Corpora from Movie and TV Subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Josef R. Novak, Nobuaki Minematsu, and Keikichi Hirose. 2016. Phonetisaurus: exploring grapheme-to-phoneme conversion with joint n-gram models in the WFST framework. *Natural Language Engineering*, 22(6):907–938.

Anton Ragni, Kate M Knill, Shakti P Rath, and Mark JF Gales. 2014. Data augmentation for low resource languages. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99.

Katsuhito Sudoh, Shinsuke Mori, and Masaaki Nagata. 2013. Noise-aware character alignment for bootstrapping statistical machine transliteration from bilingual corpora. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 204–209, Seattle, Washington, USA. Association for Computational Linguistics.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: Cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 649–652, Los Angeles, California. Association for Computational Linguistics.

Morris Swadesh. 1950. Salish internal relationships. *International Journal of American Linguistics*, 16(4):157–167.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 674–680, Beijing, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Ekaterina Vylomova, Jennifer White, Elizabeth Salesky, Sabrina J. Mielke, Shijie Wu, Edoardo Ponti, Rowan Hall Maudslay, Ran Zmigrod, Joseph Valvoda, Svetlana Toldova, Francis Tyers, Elena Klyachko, Ilya Yegorov, Natalia Krizhanovsky, Paula Czarnowska, Irene Nikkarinen, Andrej Krizhanovsky, Tiago Pimentel, Lucas Torroba Hennigen, Christo Kirov, Garrett Nicolai, Adina Williams, Antonios Anastasopoulos, Hilaria Cruz, Eleanor Chodroff, Ryan Cotterell, Miikka Silfverberg, and Mans Hulden. 2020. The SIGMORPHON 2020 Shared Task 0: Typologically diverse morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*.

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1530–1537, Florence, Italy. Association for Computational Linguistics.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2020. Applying the transformer to character-level transduction.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575.