Learning Multilingual Morphological Inflections for Use in Machine Translation

Dimitrios Haralampopoulos – CS584

December 10th, 2024

1 Introduction

The SIGMORPHON workshop, a long-running intercollegiate computational linguistics workshop, has been tackling the issues that come with learning morphology, phonology, and phonetics through machine learning methods. A major component of the competition is devoted to the learning of Morphological Inflection across many different target languages with the help of machine learning techniques. There have been many submissions and approaches to these tasks over the years, namely Straight-Through Gradient Estimation for Hard Attention [2], One-Size-Fits-All Multilingual Models [4], Sparse Two-headed Models for Inflection [3], and Universal Morphological Reinflection in 52 Languages [1]. My goal for this project is to create a generalizable model for learning these morphological inflections through previous examples. I hoped to achieve similar if not better results than the papers presented for the task, and evaluate the model's viability for use as a component in multilingual machine translation models, perhaps reducing confounding factors and ambiguity in translation and strengthening semantic transferability across languages. As it will become apparent later, I was largely unsuccessful in my endeavors. However, working on this task has given me a greater and more nuanced understanding of the problems associated with learning these inflections and their potential applications in Machine Translation.

2 Problem Formulation

The Inflection task has seen a wide variety of models and techniques used. This makes narrowing down the implementation to a single model difficult. However, it is important to note that the most popular models I observed in my thorough literature review for this task are attention-based Seq2Seq models. Therefore, I decided to use a Sparse Multihead Attention Seq2Seq model with added Hard Attention for this task. Seq2Seq is suitable for this task due to its widespread use in machine translation, speech recognition, abstractive summarization, and morphological inflection generation [5]. Two separate architec-

tures were trained, one Multilingual Morphological Inflection Model, and one Classifier-to-Pretrained Model. The second model was introduced in the hopes of minimizing the confounding factors present in training a multilingual model, such as vocabulary overlap and similar orthographies or writing systems.

3 Methods

3.1 Data Preparation

For the SIGMORPHON data, I downloaded the appropriate files for the six languages I have chosen to focus on from both the 2023 Inflection ST and 2017 CoNLL repositories. From the two repositories, the only shared sets of data were the covered test data, dev data (which I am using for validation), and train data. For each language, I merged the 2023 and 2017 files for each of the shared sets and ensured that only unique entries were kept in the combined sets. These combined sets are denoted by the language_tag_all.tsv files in each language directory. These were then further combined to make unified multilingual test, dev, and train sets for training a single multilingual model. The data was tokenized on a character-level for the input words and their targets, and the inflections were tokenized by splitting them by ";" and creating a vocabulary from them. Since I performed character-level tokenization for the words and their targets, normal word-level tokenization representations for special tokens like ⟨bos⟩, ⟨eos⟩, ⟨unk⟩, and ⟨pad⟩ could not be used. Instead, I created single-character mappings for each of them such that the sequences would become homogeneous in length during loss calculation and metric evaluation steps.

$$\langle \text{pad} \rangle \rightarrow * \quad \langle \text{unk} \rangle \rightarrow ?$$

$$\langle \text{bos} \rangle \rightarrow \hat{} \quad \langle \text{eos} \rangle \rightarrow \$$$

The decisions for ⟨bos⟩ and ⟨eos⟩ were intuitive, as these characters are used to mark the beginning and end of words in regular expressions. The token for ⟨unk⟩ was chosen due to it being representative of an unknown character and also not being used in the corpus anywhere. The decision for ⟨pad⟩ was made because some inflections require the addition of another word (and therefore a space) or a dash and thus we cannot risk confusing the model. Therefore, we use the asterisk which is not present in the corpus to denote the padding character. It was found that the maximum sequence size is 43 characters including ⟨bos⟩ and ⟨eos⟩ and that the maximum number of inflections for any given word is 8. For training the Classifier model, only the word and inflection columns were selected, and the dataset was given a new target column with an integer value from 0-5 indicating what the ground truth language of origin is for each sample. The data was then tokenized using the same vocab builder as the Multilingual model, and fed to the model for training. For each of the pretrained models, only data matching the target language was used for training, and each have their own unique vocabs for both word embeddings and inflection embeddings.

Each of the subsets used for the pretrained models were preprocessed the same as the combined multilingual dataset.

3.2 Model Formulation

The first model that I created is a Seq2Seq model with the following important characteristics:

- Hard Attention
- Sparse Multihead Attention (two-headed)
- Encoder-Decoder Architecture

These decisions were made due to the literature review I had done, and primarily influenced by the work in Straight-Through Gradient Estimation for Hard Attention [2]. Neither version of the MI model has straight-through gradient estimation implemented for the Hard Attention mechanism, and is something I would like to try out in the future if I decide to continue working on this project.

The Encoder makes use of an embedding layer for the characters of the input word and an embedding layer for the inflections. The character embedding is then passed to a BiLSTM and the inflection embedding is passed through a linear layer. The outputs from the BiLSTM and linear layer are then concatenated and sent to the Decoder. The Decoder has another character embedding layer for the target sequences, which is fed into the Hard Attention Mechanism. The context from the Hard Attention is concatenated with the embedding output and fed into the Decoder BiLSTM, from which we get the output, the hidden state, and the cell. Sparse Multi-Head Attention is then applied to the output and is then concatenated with the context from the Hard Attention before being passed to the linear Character Predictor layer. The Decoder then returns the prediction, the hidden states, the cell, and the Attention from the Hard Attention Mechanism. In the combined model, we get our Encoder outputs and then find the first input token from the target sequences (which should be a start token). We then for each remaining token in the batch of target sequences, we feed the input into the decoder, set our output in the output tensor, and set the new input for the next loop.

The Classifier-to-Pretrained model that I created utilizes the first architecture for the pretrained models. Each of the pretrained models are trained on the samples from one of the 6 selected languages. The classification model also makes use of a regular LSTM layer, as it takes tokenized word and inflection data to produce a label between 0 and 5, each corresponding to one of the six target languages. The LSTM output from the concatenated embeddings is passed to a Fully Connected layer and finally softmaxed to produce the final model output. After the class of the input text is predicted, the function will load the corresponding pretrained MI model and the input is passed to that model to obtain the final inflection prediction.

All of the models use Adam as the optimizer, and the Normalized MI model training and Classifier model incorporate a Learning Rate scheduler. Lower

patience (2) was used for the MI model than the Classifier (5), and a greater reduction factor (0.5) was used for the MI model than the Classifier (0.1).

For computation, I used Google Colab due to its GPU capabilities which allows for fast training, testing, and evaluation (relative to my native capabilities) on corpora as large as the ones presented in the previous works. Each year of the task has a baseline for its evaluation metrics, which I used as a benchmark for evaluating the model I created. After the evaluation of the Seq2Seq model, I formulated a novel approach to integrate the model(s) into basic neural machine translation frameworks. This will all be done in Python with the help of standard machine learning libraries like Scikit-Learn and Pytorch, along with data manipulation libraries like Polars. The models can be found here in this Google Drive folder.

4 Datasets and Experiments

I made use of the 2023 SIGMORPHON InflectionST and 2017 SIGMORPHON CoNLL data repositories. The 2017 dataset is valuable due to the sheer number of languages provided (52 total), while the 2023 is the most recent dataset with both training, test, and validation splits available. I aim to train and test the model on both datasets separately, as well as combined, removing duplicates. I scored the model using standard metrics such as Accuracy, Precision, Recall, F1 Score, and BLEU score, as well against the benchmark metrics provided in the 2023 and 2024 challenges. For both the Morphological Inflection and Machine Translation tasks, I have selected 6 languages from the 60 total unique languages among the SIGMORPHON corpora, those being English, Hebrew, Spanish, Russian, Turkish, and Hungarian. These languages were selected due to all of them being present in both data repositories, as well as factors such as number of speakers worldwide and linguistic variety. There were 10,000 unique training data samples for each language for 2023, and 1,000 for 2017 (since I used the medium instead of large sets), resulting in a unified dataset of $\sim$11,000 samples per language (with duplicates removed) and 65,000 samples for the multilingual dataset. Test and validation sets were the same size, 1,000 unique samples for each of the selected years, resulting in unified datasets of $\sim$2,000 samples per language and $\sim$12,000 for the multilingual dataset. The datasets I have created can be found at the Google Drive folder link here.

5 Results

5.1 Initial Morphological Inflection Evaluation

The first iteration of the Morphological Inflection model has been completed, and I would like to report my findings from that here, as it will provide some guidance for future steps. The hyperparameters I used to train these initial models and the reported metrics from the final training epoch are as follows:

Model	Model Stats					
l=1, e=10	Epochs	Batch	Hidden	Embedding	LR	Num. Layers
	10	256	256	128	0.01	1
	Loss	Acc	Prec	Rec	F1	BLEU
	0.303	0.253	0.253	0.253	0.253	0.010
l=2, e=10	Epochs	Batch	Hidden	Embedding	LR	Num. Layers
	10	256	256	128	0.01	2
	Loss	Acc	Prec	Rec	F1	BLEU
	0.349	0.252	0.252	0.252	0.252	0.010

Table 1: Training Metrics from Initial MI Models

The loss is low compared to what they started at ($\sim$3.0), yet so is the accuracy, precision, recall, f1-score, and BLEU score. To compare, the neural baseline model from 2023 had an accuracy of 81% and the non-neural baseline had an accuracy of 69% [2]. After doing some small batch random sampling and testing the outputs, it seems the model actually captures the inflection information quite well. For example, take the following data point:

Word	Inflection	Result
^tonsurar$	V;SBJV;PST;1;SG;LGSPEC1	^tonsurara$

From the model prediction, we get tonsurara$e$$\cdots$$ and the target string is tonsurara$*$$\cdots$*. Fundamentally, the result produced by the model is correct. Any word should terminate immediately after the first $. However, due to the way that we represent the predicted string and the nature of the model, when we are checking for our metrics, these "incorrect" tokens get factored in, even though the inflection is basically correct. A way that we can artificially correct our results is to apply preprocessing to the predicted tensors such that all symbols after the first $ become * tokens regardless of their original value.

5.2 Post-Regularization Morphological Inflection Evaluation

Model	Model Stats					
l=1, e=10	Epochs	Batch	Hidden	Embedding	LR	Num. Layers
	10	256	256	128	0.01	1
	Loss	Acc	Prec	Rec	F1	BLEU
	0.329	0.252	0.252	0.252	0.252	0.045
l=1, e=20	Epochs	Batch	Hidden	Embedding	LR	Num. Layers
	20	256	256	128	0.01	1
	Loss	Acc	Prec	Rec	F1	BLEU
	0.221	0.261	0.261	0.261	0.261	0.058

Table 2: Training Metrics from Regularized MI Models

I refer to the previously described preprocessing technique rather loosely as "Regularization" of the prediction output. The "Regularized" version of the l=1, e=10 model improves only with the BLEU score, quadrupling it from the score reported in the initial version. Bumping the epochs up to 20 led to a further decrease in the final loss which is good, but had minimal impact on the remaining metrics. Perhaps this is an inherent weakness of a multilingual model? In an attempt to minimize the effects of confounding factors in multilingual models, I decided that it would be worthwhile to train a classifier model and then pretrained models for each language.

5.3 Classifier-to-Pretrained Evaluation

Training the classifier was very straightforward and gave promising results straight away. Only one model was trained for classification, with a single-layer LSTM and 10 epochs. Hidden size and embedding size remain the same as in the MI model, but instead we use a smaller batch size of 128.

Metric	Value
Loss	1.057
Accuracy	0.986
Precision	0.986
Recall	0.986

Table 3: Language Classification Model Validation Metrics

Because of the instantaneous satisfactory results from the classifier, no further classification models were trained. The remainder of the efforts went to the pretrained models for each language. All pretrained models were trained for 10 epochs with a batch size of 128, and the same embedding sizes as in the previous model architecture.

Language	Train Loss	Train Acc	Valid Loss	Valid Acc	Layers
English	0.467	0.438	0.109	0.469	1
Hebrew	0.641	0.397	0.448	0.328	2
Hungarian	0.681	0.400	0.443	0.440	4
Russian	0.617	0.242	0.432	0.238	3
Spanish	0.449	0.396	0.246	0.399	1
Turkish	0.392	0.363	0.208	0.349	3

Table 4: Losses, Accuracy, and num BiLSTM layers for pretrained MI models

The green highlights indicate the model with the best value for the given metric, and the red highlights indicate the model with the worst value for the given metric. Overall, for the pretrained models, the English one performed the best. It had the highest training accuracy and validation accuracy among the six models, as well as the lowest validation loss. The worst model goes to Russian, as it has the lowest Train and Validation Accuracies compared to the other models. Based on the proportions from the validation accuracy, the model has an accuracy of $\sim 37\%$ assuming that all the language labels are classified correctly. Once again, compared to the neural (81% acc) and non-neural (69% acc) benchmarks, this model's performance is seriously lacking [2]. However, it is important to note that it does perform better than the original multilingual MI model created earlier, highlighting the benefits of pretraining individual models for each target language to reduce confounding factors commonly encountered when training multilingual models.

6 Potential Integration into Machine Translation Frameworks

Though time did not permit for me to properly design a machine translation system with MI integrated, it is something that I would still like to discuss as I believe it is important to the future of the work. A novel approach to integrating these MI models into modern MT systems is shown in the flow chart below:

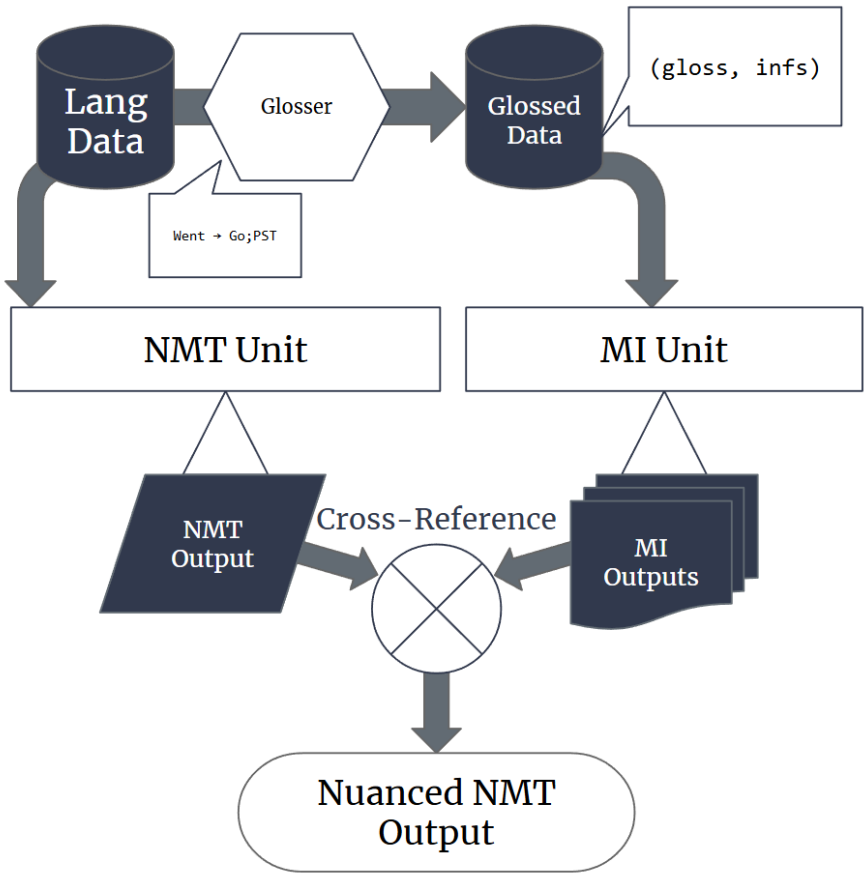

Figure 1: Basic MI integration into NMT framework

Simply put, we would like to collect or create glossed language data from our input and target languages. This is challenging because of the work required to either manually gloss the data or create an automated glosser for each of the input and target language pairs. Once we have this glossed language data, it is sent to an MI model unit and the corresponding unglossed pair is sent to the MT model. The outputs from each of the models are cross-referenced. Since not all words have morphological inflections, those shall be omitted from the cross-referencing process. This cross-referencing will hopefully allow for more nuanced MT output, taking into account cultural or semantic information that may be lost during Machine Translation. This would be a big step in the direction of universal translation, a region of NLP that has lots of room for exploration and experimentation. As much as I would have liked to dive further into this part of the problem, time constraints and resource requirements prevented me from

doing so. I do look forward to seeing what people come up with in the future for Machine Translation, especially using Multilingual models.

7 Conclusions

My primary takeaway from this project is that learning morphological inflections is really, really hard. While the problem is learnable, as is evident from the literature, it requires a significant amount of time and resources to work on effectively. My attempt at learning these inflections on my own did not go nearly as well as I thought it would. My experiments, while insightful, yielded results that were less than ideal and worse than results found in pre-established literature. The Multilingual MI model performed alarmingly badly even with the number of layers changed. Increasing the number of epochs seemed to have a positive effect on the loss and accuracy of the models, though not by much. The same can be said for the Classifier-to-Pretrained model, where the MI models didn't really seem to benefit too much from changing the number of layers either. The only really new thing I have introduced to the MT scene is the idea of using these learned morphological inflections to create more nuanced MT output. For future work, there are a number of options that can be explored. For example, using models other than RNNs for MI learning, utilizing straight-through gradient estimation for the Hard Attention module, training the models for more epochs, using the 2023 dataset in combination with the large 2017 datasets for each language (which I honestly should have just done from the beginning), and actually integrating the MI models into an NMT framework.

8 Project Management

As the sole member of this team, every step of the project was delegated to me.

References

[1] R. Cotterell, C. Kirov, J. Sylak-Glassman, G. Walther, E. Vylomova, P. Xia, M. Faruqui, S. Kübler, D. Yarowsky, J. Eisner, and M. Hulden. CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In M. Hulden, editor, *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver, Aug. 2017. Association for Computational Linguistics.

[2] L. Girrbach. Tü-CL at SIGMORPHON 2023: Straight-through gradient estimation for hard attention. In G. Nicolai, E. Chodroff, F. Mailhot, and Ç. Çöltekin, editors, *Proceedings of the 20th SIGMORPHON workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages

151–165, Toronto, Canada, July 2023. Association for Computational Linguistics.

[3] B. Peters and A. F. T. Martins. IT–IST at the SIGMORPHON 2019 shared task: Sparse two-headed models for inflection. In G. Nicolai and R. Cotterell, editors, *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 50–56, Florence, Italy, Aug. 2019. Association for Computational Linguistics.

[4] B. Peters and A. F. T. Martins. One-size-fits-all multilingual models. In G. Nicolai, K. Gorman, and R. Cotterell, editors, *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 63–69, Online, July 2020. Association for Computational Linguistics.

[5] B. Peters, V. Niculae, and A. F. T. Martins. Sparse sequence-to-sequence models. *CoRR*, abs/1905.05702, 2019.