An Ensembled Encoder-Decoder System for Interlinear Glossed Text

Edith Coates
Department of Mathematics
University of British Columbia
Vancouver, Canada
icoates1@mail.ubc.ca

Abstract

This paper presents a submission to Track 1 of the 2023 SIGMORPHON shared task on interlinear glossed text (IGT) (Ginn et al., 2023). There are a wide amount of techniques for building and training IGT models (see Moeller and Hulden, 2018; McMillan-Major, 2020; Zhao et al., 2020). We describe the system's ensembled sequence-to-sequence approach, perform experiments, and share the submission's test-set accuracy. We also discuss future areas of research in low-resource token classification methods for IGT.

1 Introduction

This paper is a system demonstration for our submission to the 2023 SIGMORPHON shared task on interlinear glossed text (Ginn et al., 2023). We focused on the closed track of the task, where only the input sentence, output gloss, and translation are provided in training data. This was more restrictive than the open track, in which more information was available, such as morphological segmentations or part-of-speech tags.

1.1 Interlinear Glossed Text

Interlinear glossed text (IGT) is a form of linguistic data annotation which highlights the grammatical properties of a corpus of text. IGT is not standardized and varies from annotator to annotator (Palmer et al., 2009), but typically uses three lines for each sentence of text. The data provided in the shared task follow the Leipzig glossing conventions (Comrie et al., 2008), in which the first line contains a transcription in an "object language," i.e. the language of study; the second line is a morpheme-by-morpheme annoatation of the sentence (called a "gloss"); and the third line is a direct translation.

(1) Ap yukwhl ha'niisgwaa'ytxw.
VER IPFV-CN INS-on-rest
But it was Sunday.

Ex. 1 shows an example in Gitksan from the task's training data. In the gloss, the functional morphemes are referred to as "grams" and the lexical morphemes are "stems," as per Zhao et al. (2020).

1.2 Related Work

Moeller and Hulden (2018) used a character-level system that combined a Support Vector Machine for recognizing grams and stems with a Conditional Random Fields labeller for assigning output grams to input characters, using a BIO-tagging convention (Ramshaw and Marcus, 1995). They also trained a character-level LSTM encoder-decoder on the BIO-tagged data.

McMillan-Major (2020) uses an ensembled system in which two CRF models focus on the source text and gloss, and translation text and gloss, respectively.

Zhao et al. (2020) use a transformer-based encoder-decoder system in which the encoder is multi-sourced: the source text and the translation are encoded separately and then combined in a single attention mechanism.

1.3 Baseline Model

The IGT shared task baseline model (Ginn, 2023) is a transformer-based token classification system. The authors found that a sequence-to-sequence model required more data to converge and performed worse when compared to the token classification approach.

2 Methods

The system was based on an encoder-decoder model using the LSTM architecture (Sutskever et al., 2014). It used ensembling and data augmentation as a method to counteract the relatively lower performance of encoder-decoder models as highlighted in the previous section. The system was implemented with Fairseq (Ott et al., 2019) and trained on a single Nvidia GeForce MX350.

Strategy	Input sequence	Output sequence
Character output	h a r i z i _ b o q n o _ ž a	r e q u e s t _ I I I - b e c o m e - T O P _ D E M 1 . S G
Token output	h a r i z i _ b o q n o _ ž a	request III - become - TOP DEM1 . SG
Stem token	h a r i z i _ b o q n o _ ž a	<stem> III - <stem> - TOP DEM1 . SG
Word-level (w=1)	b o q n o _ ž a _ <e>	DEM1 . SG
Stemmer model	t h e _ d r a g o n _ b e g g e d	r e q u e s t _ b e c o m e

Table 1: An example from the shared task's training data in Tsez, showing different preprocessing approaches.

Window size	Stem F1	Morpheme	Word
1	43%	42%	46%
2	46%	50%	**65%**
3	35%	40%	56%
1 and 2	**49%**	**54%**	64%
2 and 3	41%	47%	62%

Table 2: Development-set results for Tsez, comparing different word-level window sizes and ensembling combinations over stem F1-score, morpheme-level, and word-level accuracy. These models all use a token-level output alphabet.

Fairseq has built-in support for transformers as well as LSTMs, but the former requires more resources to train. The GPU used for this project did not have sufficient memory for training a convergent transformer model, and so the LSTM architecture was chosen instead.

2.1 Representing target glosses

The source language text was represented as a sequence of characters, and we experimented with several approaches for representing the gloss as a target alphabet. Initially, the output gloss was also represented as a sequence of characters. Later, we used a token-based output alphabet. See Table 1 for examples.

The shared task dataset includes translated stems in its glosses. We experimented with representing the stems with a special token instead, and a model for generating stems from the translation, but chose to use the token-based output with the original stems in the final results. Development-set results can be found in Table 3.

2.2 Word-level training examples

Instead of giving the system an entire sentence to gloss as one example, the system was trained with word-level examples, which included tokens on either side of the "target" word for added context. Since the output gloss contains the same number of tokens as the input sentence, training and inference can be performed on the word-level, and sentence-level results can be created from a simple concatenation of word-level results. The number of tokens on either side of the "target" became a hyperparameter, and we found that a word-window of two tokens on either side gave the best results for a single model. See Table 2 for results.

Output format	Stem F1	Morph.	Word
Characters	38%	44%	53%
Tokens	**49%**	**44%**	**64%**

Table 3: Development-set results for output formats for Tsez training data. Results for the stemmer and a special stem token are not included. Both systems use an ensemble of window size 1 and 2 word-level models.

2.3 Ensembling and voting

The final form of the system was a combination of a model trained on a window size of one, and another trained on a window size of two. During inference, Fairseq provides a negative log-likelihood (NLL) score for the model's predictions. A final output token was chosen by finding the smallest NLL score for either model's predictions. Figure 1 depicts the ensembling and voting process.

2.4 A model for predicting stems

We experimented with an additional sequence-to-sequence model for generating stems using the gloss and the translation text. The full translation was used as an input sequence, and just the stems from the gloss would be used as an output sequence. The input and output sequences were represented with a character-level alphabet.

The system used a simple technique for adding stems to the final model predictions: During the combination of word-level results into sentence-level outputs, the system replaced the special stem tokens with predictions from the stemmer model in the order that each sentence-level stemmer result was generated. If the model generated too many stems, the rightmost outputs would be left out, and

Word-level accuracy	**arp**	**ddo**	**git**	**lez**	**ntu**	**nyb**	**usp**	**AVG**
This submission	56%	74%	7%	66%	71%	77%	67%	60%
Baseline	71%	73%	17%	50%	42%	5%	72%	47%
Best other result (per language)	**79%**	**81%**	**21%**	**79%**	**81%**	**85%**	**73%**	**71%**
Morpheme-level accuracy								
This submission	45%	64%	9%	40%	37%	73%	56%	47%
Baseline	44%	51%	8%	42%	18%	14%	57%	34%
Best other result (per language)	**78%**	**73%**	**12%**	**62%**	**56%**	**87%**	**70%**	**63%**

Table 4: Test-set results of the shared task across all languages.

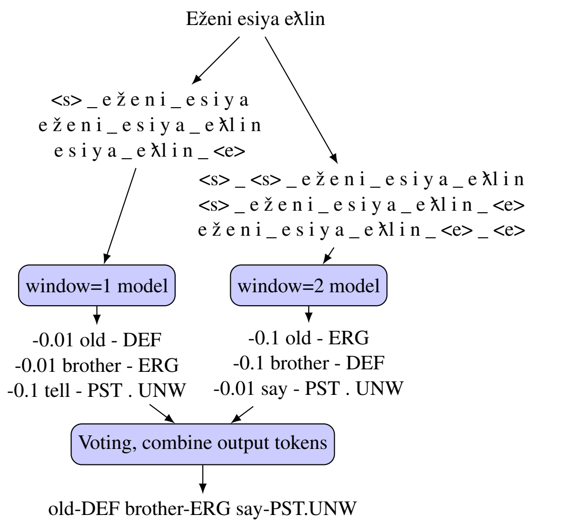
Eženi esiya eƛlin
<s> _ e ž e n i _ e s i y a
e ž e n i _ e s i y a _ e ƛ l i n
e s i y a _ e ƛ l i n _ <e>
<s> _ <s> _ e ž e n i _ e s i y a _ e ƛ l i n
<s> _ e ž e n i _ e s i y a _ e ƛ l i n _ <e>
e ž e n i _ e s i y a _ e ƛ l i n _ <e> _ <e>
window=1 model
window=2 model
-0.01 old - DEF
-0.01 brother - ERG
-0.1 tell - PST . UNW
-0.1 old - ERG
-0.1 brother - DEF
-0.01 say - PST . UNW
Voting, combine output tokens
old-DEF brother-ERG say-PST.UNW

Figure 1: A diagram of the system's approach to word-level training and voting, with an example from Tsez and hypothetical NLL scores and word-level predictions.

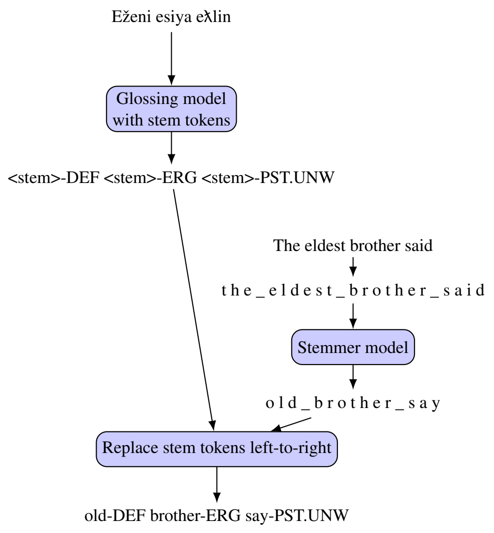
Eženi esiya eƛlin
Glossing model with stem tokens
<stem>-DEF <stem>-ERG <stem>-PST.UNW
The eldest brother said
t h e _ e l d e s t _ b r o t h e r _ s a i d
Stemmer model
o l d _ b r o t h e r _ s a y
Replace stem tokens left-to-right
old-DEF brother-ERG say-PST.UNW

Figure 2: A diagram of the system with a model that uses the translation to predict stem tokens. The glossing model could be a single word-level model or an ensemble like in Figure 1.

if there were too few, at least one special stem token would remain. Figure 2 represents a glossing system working with the stemmer model.

This stemmer model was prototypical and we found that it did not have an effect on overall performance or stem F1 score.

2.5 Evaluation

We used the shared task baseline model's evaluation script, which calculates a variety of metrics, including an overall BLEU score, stem F1, precision, and recall, as well as word- and morpheme-level accuracy.

3 Results

The final system consisted of two word-level sequence-to-sequence models, trained with a word window size of one and two, respectively. The input alphabet consisted of characters and the output alphabet was token-level. The models were trained with an inverse square root learning rate scheduler, early stopping, and the Adam optimizer. Models trained on all languages except for Gitksan used a batch size of 128. Since the Gitksan training dataset was just 31 examples long, it used a batch size of 64 instead.

See Table 4 for results across all languages in the shared task training data.

3.1 Analysis

For most of the languages, the system performed better than the baseline in terms of word-level and morpheme-level accuracy. However, the relative performance varied by language: the system's word-level accuracy for Arapaho is 15% lower than the baseline, while the same metric for Nyangbo is 72% higher.

We hypothesized that these results could have been caused by differences in morpheme-to-word ratios across the languages. Since the system was trained on word-level examples, a lower ratio would suggest longer sequences for training — Wu et al. (2021) points out that transformer models perform better than RNNs on longer sequences.

For each training dataset, we calculated the average morpheme-to-word ratio and found that Nautugu and Uspanteko have the joint highest ratios of 0.83, while Arapaho and Nyangbo are lower with 0.72 and 0.75, respectively. The language with the lowest ratio was Tsez, at 0.6.

There seems to be a weak trend: The model under-performed or was at par with the baseline for languages with low ratios. For languages with a higher ratios, the model performed better than the baseline, with an exception of Uspanteko.

From this analysis, we conclude that training data size and morpheme-to-word ratio alone cannot explain the model's under-performance for Arapaho and over-performance for Nyangbo.

4 Conclusion

This was a system demonstration of our submission to the 2023 SIGMORPHON shared task on interlinear glossed text. While the system was not the best-performing of all the submissions, it nonetheless performed consistently better than the baseline model in terms of word- and morpheme-level test-set accuracy.

The system was relatively inexpensive to train, as it was built on a single CUDA-enabled laptop. This could be an advantage of the LSTM-based architecture: when Wu et al. (2021) introduced the transformer architecture to character-level transduction tasks, the authors noted that transformer-based performance depended on finely-tuned hyperparameters and longer training times.

However, non-encoder-decoder systems for the ILG task still show lots of promise, especially for small datasets. Further research can be done to examine the effect of ensembling and data augmentation on CRF or LSTM-based token classification systems.

More work can be done on the stem generation system as well: a linguist-created inflectional database like the one described in Oliver et al. (2022) could algorithmically recognize stems and look up translations. Also, an upstream word alignment model, such as one of the IBM models described in Brown et al. (1993), could help with the construction of a stemmer system.

We hope this demonstration will lead to future work on low-resource systems for automatic ILG, in terms of both computation and dataset size.

5 Limitations

Due to time constraints, it was not possible to perform a satisfactory grid search on the large combinations of training hyperparameters, preprocessing techniques, and stem approaches. It is possible that a more optimal system is possible, but we were unable to find it.

6 Acknowledgements

Special thanks to Miikka Silfverberg, Naomi Liu, the organizers of the shared task, and the anonymous reviewers.

References

Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, Robert L Mercer, et al. 1993. The mathematics of statistical machine translation: Parameter estimation.

B. Comrie, M. Haspelmath, B. Bickel, and Max Planck Institute for Evolutional Anthropology. 2008. *The Leipzig Glossing Rules: Conventions for Interlinear Morpheme-by-morphene Glosses*. Max Planck Institute for Evolutionary Anthropology.

Michael Ginn. 2023. Sigmorphon 2023 Shared Task of Interlinear Glossing: Baseline Model.

Michael Ginn, Sarah Moeller, Alexis Palmer, Anna Stacey, Garrett Nicolai, Mans Hulden, and Miikka Silfverberg. 2023. Findings of the SIGMORPHON 2023 Shared Task on Interlinear Glossing. In *Proceedings of the 20th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics.

Angelina McMillan-Major. 2020. Automating Gloss Generation in Interlinear Glossed Text. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 355–366, New York, New York. Association for Computational Linguistics.

Sarah Moeller and Mans Hulden. 2018. Automatic Glossing in a Low-Resource Setting for Language Documentation. In *Proceedings of the Workshop on Computational Modeling of Polysynthetic Languages*, pages 84–93, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Bruce Oliver, Clarissa Forbes, Changbing Yang, Farhan Samir, Edith Coates, Garrett Nicolai, and Miikka Silfverberg. 2022. An Inflectional Database for Gitksan.

In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6597–6606, Marseille, France. European Language Resources Association.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Alexis Palmer, Taesun Moon, and Jason Baldridge. 2009. Evaluating Automation Strategies in Language Documentation. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 36–44, Boulder, Colorado. Association for Computational Linguistics.

Lance Ramshaw and Mitch Marcus. 1995. Text Chunking using Transformation-Based Learning. In *Third Workshop on Very Large Corpora*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2021. Applying the Transformer to Character-level Transduction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1901–1907, Online. Association for Computational Linguistics.

Xingyuan Zhao, Satoru Ozaki, Antonios Anastasopoulos, Graham Neubig, and Lori Levin. 2020. Automatic Interlinear Glossing for Under-Resourced Languages Leveraging Translations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5397–5408, Barcelona, Spain (Online). International Committee on Computational Linguistics.