Glossy Bytes: Neural Glossing using Subword Encoding

Ziggy Cross[1*] **Michelle Yun**[1*] **Ananya Apparaju**[2] **Jata MacCabe**[2]
Garrett Nicolai[3] **Miikka Silfverberg**[3]
University of British Columbia
[1]{zcross,bibianna}@student.ubc.ca
[2]{ananya.apparaju,jatamaccabe}@gmail.com
[3]{garrett.nicolai,miikka.silfverberg}@ubc.ca

Abstract

This paper presents several subword-modelling-based approaches to interlinear glossing for seven under-resourced languages as a part of the 2023 SIGMORPHON shared task on interlinear glossing (Ginn et al., 2023). In an interlinear glossed text (IGT), each line of the original text is paired with one or more corresponding lines which encode the underlying grammatical structure. While expert annotated glossed text is especially valuable for the study of low-resource languages in both theoretical linguistics and natural language processing, generating high-quality glossed data is expensive and time-consuming. Therefore, approaches which aim to automatically or semi-automatically generate glossed data can be valuable for linguistic research. We experiment with various augmentation and tokenization strategies for both the open and closed tracks of data. We found that while subword models may perform well for greater amounts of data, character-based approaches remain competitive in their performance in lower resource settings.

1 Introduction and Motivation

Subword[1] representations can leverage the compositional nature of input words to model the morphology of a language. Approaches that treat words as atomic units have limitations when handling morphologically rich languages (Ling et al., 2015), where words may be composed of several meaningful morphemes (which, in turn, are composed of characters). Another limitation of the word-level approach is its inability to handle out-of-vocabulary (OOV) words. When data is scarce and many test words are absent from the training set, generic OOV handling (i.e. <UNK> tagging) is especially problematic. Recent strategies for OOV handling in neural machine translation include using pre-trained contextualized word embeddings (Lochter et al., 2020) or exploiting external data (Ngo et al., 2019). However, these methods are often domain-specific and may be unrealistic in a truly low-resource setting.

In such scenarios, models capable of learning relationships between orthographically similar sequences (Ling et al., 2015) may be especially valuable for disambiguating rare and unseen words, as there is often overlap between an OOV word (e.g. *desktop*) and those present in the vocabulary (e.g. *desk, top*). A drawback (Plank et al., 2016) of character-level representations lies in the non-trivial relationship between word forms and their meanings. Subword models may represent a compromise between characters, which are semantically void, and word-level representations. Indeed, byte-pair encoding (BPE) (Sennrich et al., 2016a) can effectively handle rare and unknown words in neural machine translation, particularly when a word-level translation may be derived from the translation of word-internal units.

Throughout this paper, we examine several approaches to neural interlinear glossing, and our contributions are as follows:

1. We implement a *sliding-window* based data augmentation approach, drawing solely from the given training set, to improve results for unsegmented inputs (3).

2. We compare the outputs of input representations at two granularities (subword, character) across various language typologies (6.1.1).

3. We provide a quantitative error analysis of gloss tags generated at the character level (6.1.2).

4. We compare the performance of recursive and transformer models for pre-segmented inputs (6.2).

*The first two authors contributed equally.

[1]Throughout this paper, we use *character* to refer to simple character-level splitting and *subword* to refer to all other subword segmentation

Proceedings of the 20th SIGMORPHON workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 222–229

Additionally, we propose that sequence-to-sequence (seq2seq) models are a viable approach for automated gloss generation in low-resource scenarios even for the closed-track task, where systems are trained exclusively on unsegmented input sentences and glosses.

2 Related Work

Given the data-hungry nature of neural systems, many approaches for automating *low-resource* IGT generation (Moeller and Hulden, 2018; McMillan-Major, 2020) have been statistical, treating gloss generation as a token classification task where each morphologically segmented source line is mapped to its corresponding morphosyntactic descriptor (MSD). As CRFs cannot encode variable-length sequences, they do not extend to the closed-task setting.

The baseline (Ginn, 2023) for this task uses a BERT-based model to label each whitespace-separated sequence with its corresponding glossed unit. This choice in architecture is motivated by the scarcity of training data and fails to exploit orthographic regularities which lend consistent clues to the internal structures of morphologically rich grammars.

In a recent neural approach to automated gloss generation for Lezgi, Tsez, and Arapaho, Zhao et al. (2020) experimented with both word and byte-pair tokenization. While they noted that the subword model outperformed the word-level model for all languages but Lezgi, they did not systematically analyze each approach.

3 Data

The data for this shared task comes from seven low-resource languages from various language families. Some languages in the set include a large number of training examples, while others contain very few.[2] All languages have original texts (orthographic representations) and gold-standard glosses. Some languages also have translated lines of text (in either English or Spanish). For the open track, all languages except Nyangbo have morphologically segmented lines, and Uspanteko has POS annotations.

The format of the data was as follows for the closed track:

- `<t>`, the orthographic representation
- `<g>`, the gold standard gloss
- `<l>`, the translation (in English or Spanish)

[2]For detailed information on the languages, see Table 1

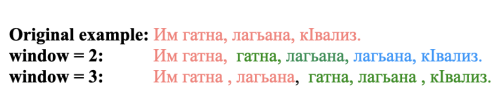

Figure 1: Sliding window augmentation

The format of the data was as follows for the open track:

- `<t>`, the orthographic representation
- `<m>`, the morphologically segmented line
- `<p>`, part of speech tags (Uspanteko only)
- `<g>`, the gold-standard gloss
- `<l>`, the translation (in English or Spanish)

3.1 Closed Track Data Augmentation

For the closed track, we used a sliding window augmentation strategy (Figure 1).

Given the training set for a language, we first define a minimum window size $lb = 1$ and a scaling factor $p = 0.5|0.7$. We count the length of each whitespace-segmented target line in the training set and find the average count c. The maximum window size ub is $c * p$. We then generate new source and target examples by segmenting each example in the training set into spans of length $lb...ub$. These spans are added back into the training set as new training instances.

Language	Original	Augmented	Total
Arapaho	39501	370058	409559
Gitksan	31	827	858
Lezgi	701	44517	45218
Natugu	791	53033	53824
Nyangbo	2100	17836	19936
Tsez	3558	238190	241748
Uspanteko	9774	103365	113139

Table 1: Overview of closed track training set

3.2 Open Track Data Representation

For the open track, all sentences were split up into individual words. Each word was represented once for every morpheme it contained, with moving

'morpheme boundaries' for each duplication (we used a <#> tag to represent this boundary). For example, the input *re-connect-ed* would be represented as follows:

<#>re <#>connect −ed
re <#>connect <#>ed
re −connect <#>ed<#>

Simplifying our input like this allowed us to represent the problem as a series of morpheme classifications, rather than a variable-length sequence output task.

Our final model only used a context size of one word, meaning each word in the input is considered independently. A larger model could allow us to include several words, or even the entire sentence, as context. A larger context could potentially allow the model to learn syntactic patterns, however, we decided this would be too computationally intensive for the shared task, as the improvements would likely be marginal.

Overall, this representation meant that our input data had as many examples as the number of morpheme tokens in given sentences.

Language	# Sents	# Words	# Morphs
Arapaho	39501	139714	251655
Gitksan	31	261	429
Lezgi	701	7029	10497
Natugu	791	10140	16341
Nyangbo	2100	8669	13778
Tsez	3558	37458	74334
Uspanteko	9774	41923	60458

Table 2: Overview of open track training set

4 Model Architecture

4.1 Closed Track

Our closed track model is a standard transformer-based sequence-to-sequence network (Vaswani et al., 2017). We use 3 layers for both the encoder and decoder, as well as 6 attention heads. Dropout is set to 0.25, and the feedforward and embedding dimensions are set to 512 and 300, respectively. The default batch size is set to 32 for both training and inference, adjusted to 8 for Gitksan and 64 for Arapaho to account for differences in the amount of available data. For training, we used PyTorch's implementation of the Adam optimizer[3], with learning rate $\gamma = 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\epsilon = 10^{-9}$. Each model was trained over 50 epochs. To prevent overfitting, we stopped the training procedure if validation accuracy did not improve for 3 consecutive epochs.

Inputs are segmented into either BPE subwords or characters. During the decoding phase, the decoder auto-regressively generates an output gloss sequence until the <end> token is reached; at translation time, the predicted token is selected via a greedy decoding mechanism.

4.2 Open Track

Our open track solution was broken up into two parts, the first being tag prediction, and the second being stem prediction. Example glosses contained a mix of MSDs and stems, and while our models would be capable of predicting both together, we decided that the two tasks should be separated due to their vastly different vocabulary sizes. For example, a language may only contain a few hundred unique tags, but several thousand stems. This separation meant we could greatly reduce the output space of each task, in turn speeding up model learning. To do this, our tag prediction model would obscure all stems by replacing them with a <STEM> tag. We could then use a more lightweight prediction model for anything our tag predictor classified as a stem.

4.2.1 Tag prediction - BiLSTM encoder

Our first approach for encoding inputs in the open track task was using a BiLSTM model. Each example was represented as a sequence of characters (or tags in the case of the morpheme boundary <#>), with each character having its own randomly-initialised embedding. Our model would retrieve the embeddings for each character and then sequentially pass them into a bidirectional LSTM network. To get the encoding of our input we took the final hidden states from each LSTM direction and concatenated them into a final context vector.

4.2.2 Tag prediction - ByT5 encoder

To improve on our BiLSTM model, we used the encoder from Google's pre-trained ByT5 model to generate our context vectors. This encoding system is much more powerful than our BiLSTM model, in part due to its higher dimensional layers, but also its pre-trained embeddings and attention

[3]Other implementations of the Adam optimizer may use α to represent the learning rate

mechanisms. When fine-tuning our models we applied multilingual training jointly on all shared task languages before fine-tuning on each individual language. This was done in order to enable some transfer learning, which may be useful for the most low-resource languages. To further this, multilingual training sets could be supplemented with data from high-resource languages to improve results in the multilingual training phase (though that may be outside the spirit of the shared task).

4.2.3 Tag prediction - Feed-forward decoder

After generating context vectors with either the BiLSTM or ByT5 encoder, we then passed the output through a feed-forward network with a single hidden layer to generate a tag prediction. The input size of the network was defined by the size of the encoder's output context vector, and the output size was defined by the vocabulary size of possible output tags observed during training time. The hidden layer size was tuned as a hyperparameter, but always remained above the output dimension.

4.2.4 Stem prediction - Most common vocab

To predict stems we used a vocabulary dictionary to map word forms seen during training with their equivalent glosses. We used a counter to keep track of the most common glosses for each morpheme and used this to replace any forms predicted to be `<STEM>` with their most common gloss. Any forms not seen during training time were replaced with `<UNK>` tags, though they could also be left as the original word form, which might improve performance on noun stems (such as names or places) where the translation and gloss might match.

5 Experiments

5.1 Closed Track - Character-level

For each language-specific model, we built separate source and target vocabularies consisting of the set of unique characters in the transcription (source) and gloss (target) lines of the training data. An early error analysis showed that OOV characters were usually non-alphabetic, so we manually added these characters to both source and target vocabularies.

Each line was split into characters and post-processed. Morpheme separators[4] were re-attached to preceding and following characters. In addition, whitespace was replaced with `#`. This step prevented the generation of ill-formed glosses with dangling separators such as '*one escape-*'.[5]

For example, the gloss *'one escape-IMPF .'* is tokenized and post-processed as follows, with segments delimited by a pipe character:

1. Original gloss:

```
one escape-IMPF .
```

2. Tokenized:

```
<start>|o|n|e|#|e|s|c|a|p|e|-|I|M|P|F|#|.|<end>
```

3. Post-processed:

```
<start>|o|n|e|#|e|s|c|a|p|e-|-I|M|P|F|#|.|<end>
```

5.2 Closed Track - BPE

We trained separate input and output BPE[6] tokenizers for each dataset, defining the maximum threshold for convergence operations given a set of characters C as $n * |C|$ Although we set $n = 16$ to avoid re-training the tokenizers at different vocabulary sizes, fine-tuning the number of merge operations is likely to yield improved results.

5.3 Open Track - BiLSTM

In our first experiments, we fed the open track data (as modelled in 3.2) into our BiLSTM encoder then feed-forward decoder and most-common-vocab stem prediction model. This performed very well and could be trained within minutes when run locally on a CPU. Our model used early stopping and would keep training only until a drop in the model's accuracy on the development was observed.

5.4 Open Track - ByT5

In our later experiments, we used our ByT5 encoder along with the feed-forward decoder and most-common-vocab stem prediction model. This model took significantly longer to train due to its much more complex architecture. After one week of training, we were unable to get it to perform better than the BiLSTM encoder model, however, we expect that its architecture should theoretically allow for a higher performance ceiling given sufficient training.

[4] Corresponding to the Leipzig Glossing Rules 2 and 4

[5] Whitespace was re-inserted and duplicate separators were removed prior to evaluation

[6] The Hugging Face implementation based on Sennrich et al. (2016b)

To assist multilingual learning, inputs to this model were prepended with a 3-character language tag, which bypassed the byte-level encoding and was treated as a special character with its own embedding. We believe this should help the model distinguish between orthographically similar languages, though further testing would be useful to determine how strong this approach is.

When training this model we used checkpointing to save the weights after each epoch. We then used an evaluation pipeline to assess the results at each checkpoint in order to determine the best model. We began fine-tuning with individual languages after 20 multilingual epochs.

6 Results & Discussion

6.1 Closed track

First, it must be noted that our approach does not appear to extend to the truly low-resource setting given its poor performance on Gitksan. Moreover, improvements in word accuracy are inconsistent, which is unsurprising given the limitations of character-level modelling discussed above. For the remaining languages, our character-level sequence-to-sequence model consistently and noticeably outperforms the baseline model for average morpheme accuracy. The only exception is Lezgi, where there is no significant difference between the morpheme accuracies. This may be due to the size of the Lezgian dataset as well as the structure of the language, but we leave this question for further investigation.

6.1.1 Character vs BPE

Both the character-level and baseline models outperformed the BPE model for all datasets apart from Arapaho; this makes sense since the generalizability of the byte-pair encoding algorithm (w.r.t. identifying rare sequences in the vocabulary) depends on the size and diversity of the training data. As we used a generic approach to training each BPE tokenizer, our results do not necessarily align with a more robust implementation of the byte-pair encoding algorithm. Although we suggest that BPE modelling is likely to be a competitive approach when more training data is available, the hands-off appeal of the character-level approach should not be ignored, especially in the context of the low-resource glossing task. If the dataset is sufficiently large, however, BPE could prove more efficient due to its compactness.

6.1.2 Qualitative Analysis

We analyze examples of predicted glosses for Arapaho (3). In the positive examples (3.1, 3.5), the predicted and target stems are consistent in meaning, while the negative examples (3.3, 3.6) are less coherent. We offer the following observations, with the caveat that we have yet to conduct a systematic analysis:

- There might be a relationship between stem rarity and prediction coherence; that is, the less frequent a stem, the less semantic similarity between predicted and target tags.
- The character-based model might do a better job at predicting semantically related tags for morphologically complex stems.
- Misalignment errors (such as in Table 3.2), where the model fails to generate a gloss tag for each word in the transcription line, occur frequently.

6.1.3 Generalization to unseen stems

The character-level model is able to successfully predict unseen English stems (Table 3.7). When the model encounters an unknown lexeme, it seems to have learned to replicate the stem (or stem-internal constituents) to preserve meaningful elements.

Notably, Arapaho is the only language where the model learns to produce unseen English words. (Liu et al., 2018) report similar results in their character-level seq2seq translator for OOV handling in statistical machine translation: the model learns to produce novel English target words by combining previously seen subwords or transliterating complete sequences. As their system (1) is specifically designed for OOV prediction for moderate-resource languages and (2) leverages external data (bilingual dictionaries, translation tables, there could be a data threshold for stem generation. With a high tag accuracy, this could prove useful for researchers who could prioritize glossing the stems and leverage the model to generate the tags. Orthographic similarity may also play a role: in our data, Arapaho is transcribed in the Latin alphabet while Tsez, the second-largest dataset, is transcribed in the Cyrillic alphabet.

6.2 Open track

We found that the BiLSTM model performed strongest compared to preliminary training on the

Token	Original Sentence	Reference	Prediction
(1) heneenei3oobei'i3i'	Nuhu' tih'eeneti3i' heneenei3oobei'i3i'	IC.tell.the.truth-3PL	IC.true-3PL
(2) Beetbeteeenehk	Beetbeteeenehk wo'uuceh nee'eesoo'	want.to-dance-2S.SUBJ	want.to-dance-SUBJ
(3) te3ou	B Tous te3ou	sandhill.crane	tell.story
(4) ne'koxo'useet	Ne'P ne'koxo'useet	then-walk.slowly-3.S	then-slowly-slowly-3.S
(5) he'ihce'oo'eixootiin	Noh he'ihce'oo'eixootiin	NARRPAST-again-people.assemble	NARRPAST-back-people.are.gathering-0S
(6) hoowuhneniinoo'	'oh hinee 3eboosei3iihi' hoowuhneniinoo' hoowu...	IC.lots.of.things-0S	NEG-too.man-0S
(7) sycamore	nuhu' sycamore huunohootin	sycamore	sycamore

Table 3: Error analysis for Arapaho (character level modelling)

	Char			Byte			Baseline		
Language	BLEU	Word	Morph	BLEU	Word	Morph	BLEU	Word	Morph
Arapaho	0.61	0.72	0.74	**0.65**	**0.74**	**0.76**	0.418	0.701	0.519
Gitksan	0.00	0.04	0.06	0.00	0.02	0.09	**0.045**	**0.291**	**0.163**
Lezgi	0.44	0.52	0.48	0.30	0.28	0.32	**0.520**	**0.557**	**0.492**
Nyangbo	0.72	0.79	**0.82**	0.68	0.73	0.76	**0.742**	**0.824**	0.782
Tsez	**0.72**	**0.77**	**0.76**	0.63	0.65	0.65	0.578	0.721	0.529
Uspanteko	**0.63**	**0.71**	**0.70**	0.54	0.63	0.63	0.538	0.703	0.655
Natugu	**0.52**	**0.58**	**0.51**	0.42	0.35	0.38	-	-	-

Table 4: Closed track evaluation results

ByT5 architecture. We believe that additional fine-tuning for individual languages on the ByT5 model would improve performance enough to beat the BiLSTM model results, however, the model's computational complexity meant we didn't have the time required to train the model to this level. Our BiLSTM model was able to outperform the baseline when used on Gitksan, Lezgi, Tsez, and Uspanteko. It was not able to beat the baseline on Arapaho and Nyangbo.[7]

The biggest strength of the BiLSTM model over the ByT5 model was its significantly lower computational complexity. We found that the BiLSTM model could be trained in minutes with a consumer-grade CPU, and the transformer-based ByT5 model took up to several hours to train a single epoch when using a research-grade GPU, which is what likely led to its worse performance in our final results. The BiLSTM architecture is impressively competent for the interlinear glossing task, and its short training time with strong performance shows that this model is a strong contender, even against the vastly more complex transformer model.

6.3 Future Work

Further research could investigate the relationship between morphological attributes (such as morpheme-to-word ratio) and the extent to which neural models can leverage compositional cues in orthographic sequences.

[7]For details on model performance, see Table 5

	BiLSTM			ByT5			Baseline		
Language	BLEU	Word	Morph	BLEU	Word	Morph	BLEU	Word	Morph
Arapaho	0.76	0.84	0.90	0.64	0.76	0.84	**0.792**	**0.854**	**0.911**
Gitksan	0.13	**0.38**	**0.52**	0.07	0.16	0.37	**0.142**	0.250	0.300
Lezgi	**0.71**	**0.83**	**0.87**	0.69	0.82	0.85	0.420	0.326	0.501
Nyangbo	0.60	0.72	0.81	0.23	0.50	0.58	**0.784**	**0.847**	**0.892**
Tsez	**0.75**	**0.79**	**0.88**	0.45	0.56	0.72	0.686	0.742	0.850
Uspanteko	0.64	**0.77**	**0.82**	0.39	0.66	0.69	**0.649**	0.759	0.813
Natugu	**0.84**	**0.89**	**0.93**	0.43	0.76	0.84	-	-	-

Table 5: Open track evaluation results

Additionally, we would like to implement some utilisation of the translation track. For example, this could be used to help resolve unknown ("<UNK>") tokens, by checking which lemmas have already been predicted and selecting the most likely of the rest. In our current approach, this potentially valuable data is left unused.

Another approach we would like to try for resolving unknown tokens is choosing a 'nearest neighbor' vocabulary item to replace any unknown stems. This would help mitigate the impact of misspellings and input noise, which our current stem prediction model (used in the open track models) is not robust to.

When predicting output tags in our open track models, which pair an encoder with a feed-forward decoder, an improved approach could involve feeding previous tag predictions into the model. This would allow us to model some (unidirectional) relationships between words without increasing the context size. This could easily be done by replacing the feed-forward decoder with a recursive decoder, such as a unidirectional LSTM or GRU.

For our BiLSTM model, implementing a patience mechanism into the early stopping might also allow for some improvements in performance and prevent underfitting.

We would also like to train our ByT5 model on a larger transformer architecture. We are currently using Google's ByT5-small model, as implemented on Hugging Face, however, there are several larger models that could easily be swapped in. In addition to this, models could be trained on larger contexts in order to learn inter-token patterns. For example, we could use a context of 3 words (the target word plus one word on either side) instead of only giving the model one word at a time.

For low-resource languages, we would also like to try Good-enough Compositional Augmentation (Andreas, 2020), as well as other data augmentation strategies. We believe this would be beneficial for models with very few training examples, such as Gitksan.

On closed-track tasks, we suggest that using a beam search decoding algorithm may yield better results than the current greedy decoding implementation, which has limited performance, particularly with longer sequences.

7 Conclusion

In this paper, we explored the potential of using subword representations in grammatical gloss-generating models to 'learn' the morphological patterns of a low-resource language. At a byte-pair level, we found this strategy to be competitive but dependent on the amount of training data available. As such, the byte-pair tokenized model performed the best for Arapaho (the dataset with the most tokens). We recognized that there might be more robust ways to implement this tokenization for languages with fewer tokens, and attributed some of the underperformance of the model in other languages to our generic tokenization strategy. We found that at a character level, even with no pre-augmentation and fewer tokens, the model delivered impressive results. We propose that the character level modelling approach excels in terms of both accessibility and performance in this setting.

Acknowledgements

We would like to thank Hariharavarshan Nandakumar and Jayathilaga Ramajayam for their help in early experiments using CRF modelling.

We would like to thank Farhan Samir for his help in preparing the ByT5 model.

This research was supported in part through computational resources and services provided by Advanced Research Computing at the University of British Columbia.

References

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

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.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Nelson F. Liu, Jonathan May, Michael Pust, and Kevin Knight. 2018. Augmenting statistical machine translation with subword translation of out-of-vocabulary words.

Johannes V. Lochter, Renato M. Silva, and Tiago A. Almeida. 2020. Deep learning models for representing out-of-vocabulary words.

Angelina McMillan-Major. 2020. Automating gloss generation in interlinear glossed text. *Proceedings of the Society for Computation in Linguistics*, 3.

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.

Thi-Vinh Ngo, Thanh-Le Ha, Phuong-Thai Nguyen, and Le-Minh Nguyen. 2019. Overcoming the rare word problem for low-resource language pairs in neural machine translation. In *Proceedings of the 6th Workshop on Asian Translation*. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. *arXiv preprint arXiv:1604.05529*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

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.

A Hyperparameters

For the model using ByT5 encoding, the output decoder (4.2.3) used one hidden layer of size 1024, which was large enough to encompass the approximately 700 size tag vocabulary.

For the model using BiLSTM encoding, the hidden layer size changed based on the size of the language-specific tag vocabulary (as this model did not use any multilingual training).

No other hyperparameters were optimised for the open track.

All of our model training and prediction code for the shared task can be accessed on GitHub at `https://github.com/michelleyun98/sigmorphon2023-IGT`.