Exploring Neural Architectures And Techniques For Typologically Diverse Morphological Inflection

Pratik Jayarao, Siddhanth Pillay, Pranav Thombre, Aditi Chaudhary
{pjayarao,spillay,pthombre,aschaudh}@cs.cmu.edu

Abstract

Morphological inflection in low resource languages is critical to augment existing corpora in Low Resource Languages, which can help develop several applications in these languages with very good social impact. We describe our attention-based encoder-decoder approach that we implement using LSTMs and Transformers as the base units. We also describe the ancillary techniques that we experimented with, such as hallucination, language vector injection, sparsemax loss and adversarial language network alongside our approach to select the related language(s) for training. We present the results we generated on the constrained as well as unconstrained SIGMORPHON 2020 dataset (Vylomova et al., 2020). One of the primary goals of our paper was to study the contribution varied components described above towards the performance of our system and perform an analysis on the same.

1 Introduction

Morphological inflection is the process of generating varied representations of words based on several linguistic properties(gender, tense,etc). Inflections of words retain their core meanings, however they differ in their word structure. As mentioned by (Faruqui et al., 2015), morphological inflections can be generated from the root word through two primary methods: concatenative measures and non-concatenative measures. In the case of concatenative measures, suffixes and prefixes are added to the original word to generate various inflectional forms of the word. Non-concatenative inflectional forms are generated by changing the basic structure of the original word. The generation of inflectional forms of a word has proved to be an asset in a wide array of NLP tasks.

Prominent languages like English, Spanish, French, etc. have large corpora that can be utilised to train large scale machine learning applications. However there are several languages in today's world that are not as well documented. These languages are termed as "low resource" languages. Morphological inflection has proven to be an effective tool to augment the datasets of "low resource" languages, so that they corpora can be better modeled using NLP techniques.

To this end, several studies have been done on morphological inflection on monolingual high resource settings, such as in the SIGMORPHON 2016, 2017, 2018 challenges. However, the low resource setting has been extensively studied in the SIGMORPHON 2019 and 2020 shared task (Vylomova et al., 2020). The data in these tasks consists of the form of [I, O, T], where I, O, T stand for input lemma, inflected form and tags respectively. The inflected form is essentially the inflected form of the input lemma upon applying the tags specified by T.

In this paper, we present an overview of the various techniques that we implemented to perform the task of Morphological Inflection in both the constrained and unconstrained settings. We start by describing the different models that we experimented with to improve our performance on this dataset. Furthermore, we describe hallucination, sparsemax/sparseloss, adversarial language network and language vector injection techniques that we prototyped to improve the performance of our system. In order to understand the impact of each component on the performance of the system, we perform a detailed analysis on the influence of these techniques on varied set of languages.

2 Related Works

In recent years there has been an increase in work in the field of extremely low resource languages. The work recent work done in the field of mor-

phological inflection can be divided into two main categories: Non-neural approaches and Neural approaches.

The non-neural based approach proposed by (Cotterell et al., 2017) has two stages, alignment and rule generation. A prominent work that combines neural and non-neural approaches is that of (Wu and Cotterell, 2019), where they seek to incorporate monotonicity as an inductive bias in their approach and develop a cubic-time based dynamic programming approach with a greedy decoding scheme. The paper hypothesizes that the monotonic attention-based models perform worse off because they were not jointly trained to incorporate the alignments.

Neural based approaches have recently outperformed the non neural based approaches. (Faruqui et al., 2015) introduces a neural network based strategy, for the task of morphological inflection generation, for languages that are morphologically rich. The authors introduce an encoder decoder based architecture which makes use of character level embeddings. (Çöltekin, 2019) on the other hand adopts the idea from transition-based parsers where the aim is to predict the parsing actions (copy, replace(c), insert(c), delete) in a given state of parser. In the recent years attention based models have gained huge popularity in Natural Language Processing tasks. (Peters and Martins, 2019) introduce a model inspired by sparse sequence to sequence models with a two-headed attention mechanism. The attention and output distributions are computed with Sparsemax function and Sparsemax loss is optimized. (Anastasopoulos and Neubig, 2019) introduce yet another attention based model which is trained on multiple languages and tries to leverage the knowledge learnt on high resource languages for low resource languages. The authors propose a novel two-step attention decoder architecture. Moreover, (Anastasopoulos and Neubig, 2019) augment low resource datasets with data hallucination.

3 Methodology

We implemented four variants of Sequence to Sequence architectures to tackle the problem of morphological injection. We primarily utilize LSTM and Transformers (Vaswani et al., 2017) to construct our models. Additionally we experimented with four techniques Hallucination (Anastasopoulos and Neubig, 2019), Sparse Max-Loss (Peters and Martins, 2019), Language Adversarial Network (Anastasopoulos and Neubig, 2019)(Chen et al., 2019) and Language Vector Injection (Littell et al., 2017).

3.1 LSTM Encoder Decoder (LSTM)

We prototyped an elementary LSTM sequence to sequence model. We incorporated two LSTM encoders with each individual encoder taking the input as the Lemma and Tags respectively. Furthermore, we implemented two separate attention heads one for the encoded representation of the input Lemma and one for the encoded representation Tags. The decoder was input the context vector and the LSTM representations with the inflected form being generated in an autoregressive manner. The architecture can be seen in Figure 1.

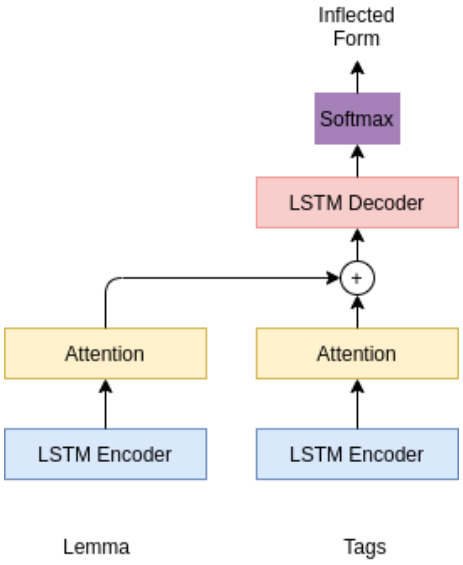

Figure 1: Lstm Encoders Decoder (LSTM)

3.2 Transformer Encoders LSTM Decoder (TELD)

Sequence Translation models such as Recurrent Neural Networks or Convolutions Neural Networks are typically trained in an encoder decoder configuration. Recently, the use of attention has shown improvement in the performance of such models. Thus we replace the LSTM encoders in the previous modules with Transformer encoder (Vaswani et al., 2017). The rest of the architecture is the same as presented in the LSTM model. We generate the output sequence using a LSTM Decoder. The structure of the architecture is shown in Figure 2.

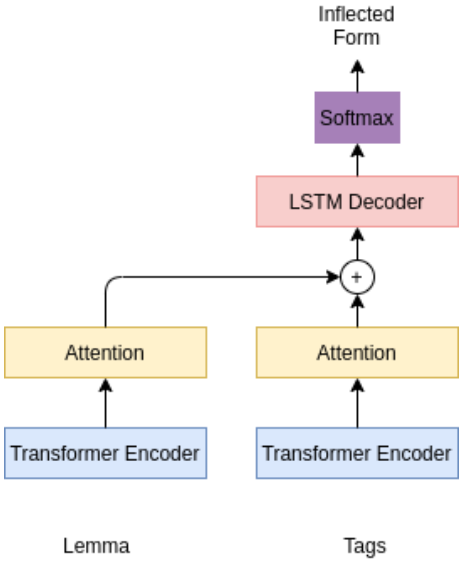

Figure 2: Transformer Encoders LSTM Decoder (TELD)

3.3 Transformer Encoders Transformer Decoder (TETD)

We further replace the LSTM Decoder with a Transformer Decoder. The two Transformer Encoders separate disparate encoder representations for the Lemma and Tags respectively. We concatenate the representations generated by the two Transformer Encoders and feed it to the output Decoder. Since the Transformer Decoder inherently has a multi-head attention layer, we remove the explicit attention over the encoders. An outline of the model can be seen in Figure 3.

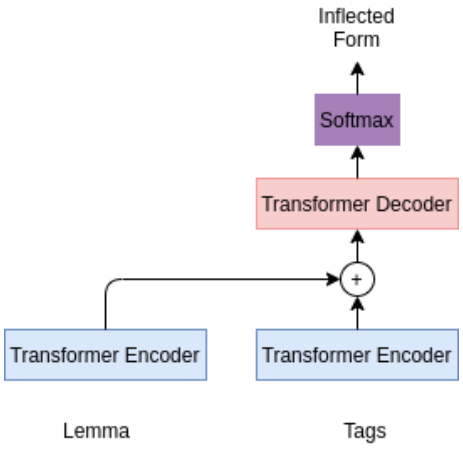

Figure 3: Transformer Encoders Transformer Decoder (TETD)

3.4 Joint Transformers (TJ)

The final architecture we implement is an end-to-end Transformer model. We concatenate the Lemma and the Tag and feed it to the Transformer. The Transformer encoder learns a joint representation for the Lemma and Tag. And the decoder generates the required output. A representation of the architecture can be seen in Figure 4.

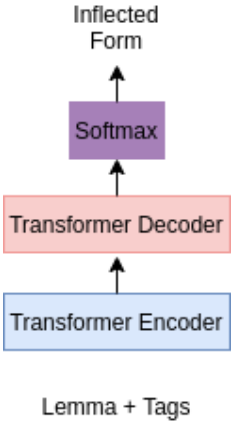

Figure 4: Joint Transformers (TJ)

3.5 Hallucination (HALL)

(Anastasopoulos and Neubig, 2019) incorporated Hallucination techniques and observed a performance boost in their system. Since the data for low resource language is scarce, the distribution learnt by the model for the language doesn't match the true distribution. To help alleviate this problem, we use this data augmentation technique. In this process each part is considered as a "stem", characters inside the region are randomly substituted with other characters without changing the overall length. A detailed explanation can be found in (Anastasopoulos and Neubig, 2019).

3.6 Sparse-Max and Sparse Loss (SPARSE)

Output vocabulary space can be potentially large with some of the characters not being used as frequently in the language. Sparsemax assigns exactly zero attention weight to irrelevant source tokens and implausible hypotheses and is shown to return sparse posterior distributions. This makes the model output more interpretable and can also help to filter out large output spaces. SparseLoss is the loss typically associated with Sparsemax and is known to be computationally very feasible. The incorporation of Sparse-max and Sparse loss in a manner similar to that of (Peters and Martins, 2019)

can be seen in Figure 5.

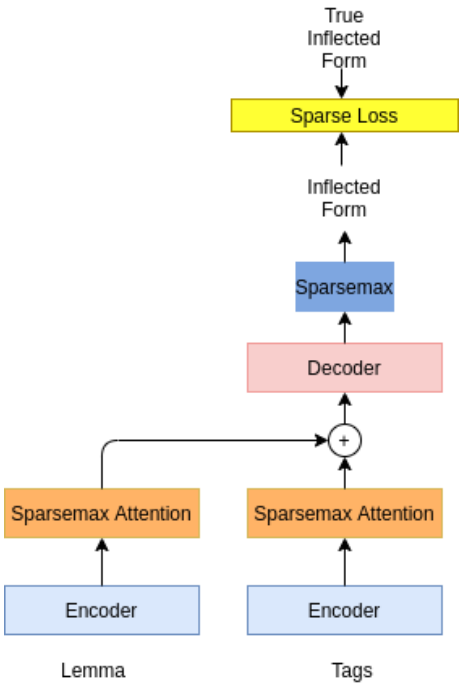

Figure 5: Sparse-Max and Sparse Loss

3.7 Adversarial Language Network (ADV-LANG)

In multilingual setting and in particular trying to transfer knowledge between related language(s) and a target language it is sometimes useful to learn language agnostic representations. Thus we implement a Language Adversarial Network which encourages the same. We extract the representations generated at the first time step and the last time step by the Lemma encoder and concatenate these representations. This representation is then passed through a linear layer and a softmax layer which produces a prediction for the Language. We then reverse the gradient while training. An illustration of the same can be seen in Figure 6.

3.8 Language Vector Injection (LVI)

(Tsvetkov et al., 2016) show that vectors which encode information about the genetics of language outperform simple one-hot representations. The lang2vec released by (Littell et al., 2017) represent languages using rich typological, geographical and phylogenetic vectors. These vectors mainly consist of binary language facts pertaining to the language such as if negation precedes a verb, is it represented as a suffix, if a language is part of Germanic family, etc. with the value of each of

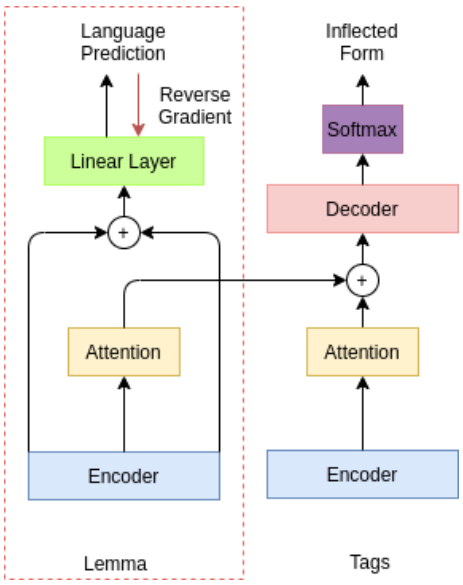

Figure 6: Adversarial Language Network

these facts represented between [0.0, 1.0]. We propose that the injection of these rich vectors into our model may increase the performance for low resource languages where training data is scarce and all round characteristics of a language cannot be learnt just from the training data.

To integrate language vectors we first extract the lang2vec vector for a particular language. We pass it through a two layer dense neural network. This provides us a compact representation for the vector. We then concatenate this representation with the output representation generated by the decoder. We then pass this through a softmax layer and the output character is evaluated. The integration can be seen in figure 7.

Furthermore we conducted a set of experiments by initializing the hidden and cell states of the (LSTM) model with language vectors but did not see promising results.

3.9 Selecting Related Language(s) for given Target Language

To select the related language(s) and target language pairs for training, we utilised the precomputed feature distance present in the Lang2Vec library(Littell et al., 2017). This distance is the cosine distance between the vectors obtained by combining the Geographical, Phonological, Syntactic, Inventory and Genetic features present in the Lang2Vec database. We assume that this distance accurately represents a metric to measure the similarity between language pairs.

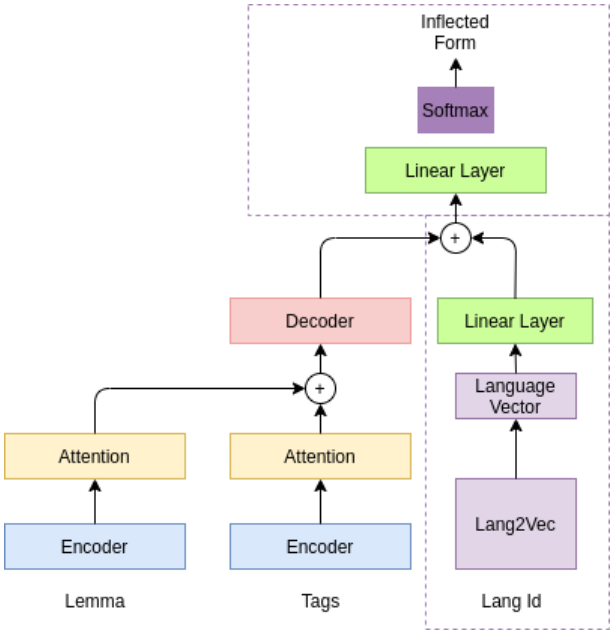

Figure 7: Language Vector Injection

4 Experimental Results

We performed our experiments on the data provided in the SIGMORPHON 2020 shared task. The dataset consists of 90 languages. The data for each language consisted of triplets in the {input, output, tags} format, where the 'output' was the output word generated after applying the morphological tags as specified by 'tags' on the 'input' word. The languages we split into two halves.The first half consisted of 45 languages development languages and the latter half consisted of 45 surprise languages.

We made submissions on all 90 languages for two different settings, unconstrained and constrained. For the unconstrained setting we trained our model in a cross-lingual manner. To complement the languages with a low number of training examples we included genetically close languages to augment the training process as explained above. For the constrained setting we restricted our training to only a single language.

As explained above we implemented various models such as (LSTM), (TELD), (TETD) and (TJ) augmented with techniques such as (HALL), (ADV-LANG), (SPARSE), (LVI). Since (HALL) has proven to perform better than the original setting we augment all languages with less then 10,000 training samples to a complete 10,000 training instances and thus all models and techniques presented below are built on top of hallucinated data. We present the results on a small subset of languages (due to the space constraints) on the development set (since we have results on all the models we trained on the development set) for the unconstrained and constrained settings in table 2 and table 3 respectively.

We did not experiment with hyperparameters and had a constant set of hyperparameters for all languages. We trained our models with the following hyperparameters 1. A further fine-tuning per language basis might have provided us with a more competitive score. But since one of the primary goals of our study was to understand the influence of the various components on our system we did not pursue this avenue in great detail.

We made a total of 5 submissions to the shared task: 3 in the unconstrained settings and 2 in the constrained setting. The submissions made to the unconstrained section are the top 3 ranked results we obtained on the development set and top 2 results for the constrained section.

Hyperparameter	Value
Optimizer	Adam
Initial Learning Rate	0.001
State Size	1024
Embedding Size	256
Number of Heads	4
Dropout	0.3
Batch Size	32

Table 1: Hyperparameters used for training the 4 models

5 Analysis

Our approach of generating morphological inflections, encapsulates several models namely: LSTM Encoder Decoders, Transformer Encoder LSTM Decoder(TELD), Transformer Encoder and Transformer Decoder(TETD) and Joint Transformers(TJ). To supplement these models, we have utilised additional strategies namely Adversarial Language Networks, Language Vector Injection and Sparse Max and Sparse Loss.

5.1 Analysis of Models Used

In our experiments, we saw that the Transformer based models, usually outperformed LSTM based

Target Language	Related Language(s) (ISO 639-3)	Model	L1+L2	ADV-LANG	SPARSE	LVI
Zulu	gaa,lug,aka	LSTM	0.81	0.83	0.81	0.83
		TELD	0.83	0.84	0.83	**0.86**
		TETD	0.81	0.84	0.83	0.83
Chichicapan Zapotec	azg,cly	LSTM	0.84	0.83	0.87	0.84
		TELD	0.87	**0.88**	**0.88**	**0.88**
		TETD	0.85	0.86	0.85	**0.88**
Yoloxóchitl Mixtec	gmh,ang	LSTM	0.86	**0.89**	0.87	0.88
		TELD	0.84	0.84	0.84	0.83
		TETD	0.81	0.79	0.81	0.79
Sotho	nya,dan	LSTM	**1.0**	**1.0**	**1.0**	**1.0**
		TELD	0.98	**1.0**	0.96	**1.0**
		TETD	0.94	0.96	0.90	0.96
Luganda	lin,zul,ceb	LSTM	0.90	0.90	0.90	0.89
		TELD	0.91	0.90	**0.91**	0.90
		TETD	0.82	0.83	0.82	0.82
Livonian	gmh,ang,kon,swa	LSTM	0.91	0.91	0.92	0.91
		TELD	0.92	0.92	**0.94**	0.92
		TETD	0.67	0.71	0.71	0.70
Classical Syriac	ang	LSTM	**0.94**	0.92	0.83	**0.94**
		TELD	0.92	0.91	0.93	0.94
		TETD	0.93	0.92	**0.94**	0.93
Kannada	nob	LSTM	0.79	0.78	**0.83**	0.80
		TELD	0.79	0.79	0.79	0.80
		TETD	0.77	0.75	0.79	0.57
Swiss German	mlg,dan	LSTM	0.87	0.86	0.87	0.87
		TELD	0.85	**0.88**	0.86	0.85
		TETD	0.78	0.77	0.76	0.78

Table 2: Accuracy obtained on 6 languages from the SIGMORPHON 2020 dataset in the unconstrained setting, where the languages were trained in conjunction with related language(s). Related language(s) have been presented in their ISO 639-3 code format.

models in general for most language pairs. Specifically, the Transformer encoder and LSTM decoder model showed the most optimal performance across all the language pairs. The ability of Transformer based models to capture long-distance dependencies, makes them more adept at generating inflections for words that were longer in length. This ensures that these models have a higher accuracy at the morphological inflection task as compared to standard LSTM based models. We can also observe that the joint transformer method was the least optimal method for most language pairs. We assume this is primarily because this method encodes both the input lemma and tags together. By encoding the lemma and tags together, we cannot utilise the information present in the tags to determine the next character to be generated during the decoding process.

5.2 Utility of Adversarial Language Network

As mentioned in (Anastasopoulos and Neubig, 2019), in a multi-lingual setting it is essential to ensure that the output of the encoder should be independent of the input language. This is vital in the task of morphological inflection generation for low resource languages. The primary reason behind this, is that while training inflection generation systems, low resource languages are trained with related language(s) that has a similar structure, due to paucity of training data.

In the context of our experiments, the adversarial language network was applied with each model that we trained, to ensure that the output of the encoder was language invariant. For the SIGMORPHON 2020 dataset, the use of adversarial language net-

Language	Model	SPARSE + LVI
Zulu	LSTM	0.81
	TELD	**0.86**
	TETD	0.83
Chichipan Zapotec	LSTM	0.85
	TELD	**0.88**
	TETD	0.87
Yoloxóchitl Mixtec	LSTM	**0.87**
	TELD	0.84
	TETD	0.79
Sotho	LSTM	1.0
	TELD	**1.0**
	TETD	0.94
Luganda	LSTM	**0.91**
	TELD	0.90
	TETD	0.80
Livonian	LSTM	**0.91**
	TELD	0.91
	TETD	0.82
Classical Syriac	LSTM	**0.94**
	TELD	0.93
	TETD	0.93
Kannada	LSTM	**0.80**
	TELD	**0.80**
	TETD	**0.80**
Swiss German	LSTM	**0.90**
	TELD	0.89
	TETD	0.80

Table 3: Accuracy obtained on 6 languages from the SIGMORPHON 2020 dataset in the constrained setting, where the languages were trained without using any related language(s).

work was found to be beneficial for most of the language pairs that we tested. However for some of our models, performance remained unchanged after the introduction of the adversarial language network. We believe that the reason for this static performance lies in the fact that the related language(s) and target language we chose during training already possessed high structural similarity. We hypothesize that this particular method would be highly useful in cases where the related language(s) and the target language pair differ widely in their structure.

5.3 Use of SparseMax and Sparse Loss

In the SIGMORPHON 2020 challenge, this technique was useful for the Chichicapan Zapotec,Zulu and Livonian languages. We hypothesize that this improvement in performance due to the addition of SparseMax is primarily because of the large vocabularies of these language pairs. For all the other languages that we tested, we noticed that we achieved a similar level of performance after the incorporation of SparseMax. The addition of SparseMax and Sparse loss aided the LSTM encoder-decoder models to a greater extent as compared to the Transformer based models that we proposed.

5.4 Utility of Language Vector Injection

We seek to use language vectors to improve performance for low resource languages where we find a paucity of data. These vectors contain embedded information about the language that we hope will be useful while generating inflectional forms of input lemmas. For the SIGMORPHON 2020 dataset, the use of language vectors helped us improve performance in almost all the language pairs that we tested. We believe the structural information embedded in the language vectors helped our model efficiently generate morphological inflections.

6 Future Work

Due to the time constraints we were not able to search through all combinations of the techniques that were mentioned such as LVI+SPARSE+LANG-ADV+NO-HALL and various others. Moreover, further fine-tuning the model hyperparameters for each language could have yielded better results.

Additionally multiple approaches to language vector injection can be explored. The vectors can be fed to the model at every-time step of the encoder by concatenating the input with the vectors or the decoder by concatenating the language vector to the context vector. This form of early injection of the vectors may help the system perform better. Another approach can be feeding the language vector to the system in place of the <sos> token.

The limited availability of supervised data for low resource languages makes it difficult to train the various data hungry Neural Network models. It has been shown that incorporation of unlabelled data can help improve the performance of such models and thus we propose to integrate a semi-supervised approach by learning Language Models over these low resource languages.These language models inherently contain information about the appropriate character sequences in a given language and thus provide valuable information for predict-

ing the next character in the decoding process. We propose to combine the probability generated by the language model with the with probability generated by the inflection model and learn the interpolation weights during training similar to the experimental setup of that of (Faruqui et al., 2016).The language model can be constructed using a basic recurrent model or even complex models such as BERT. (Devlin et al., 2018).

7 Conclusion

This paper presents a detailed description of the models that we implemented to undertake the "Typologically Diverse Morphological Inflection" shared task. We describe our encoder-decoder based approach using both LSTMs and Transformers. We also describe the different supporting techniques that we implemented, such as hallucination, language vector injection, adversarial language traning and sparsemax. We present a brief subset of the results for the SIGMORPHON 2020 dataset. We also delve deeper and try to present a detailed analysis of the different components of our model and their influence on the performance.

References

Antonios Anastasopoulos and Graham Neubig. 2019. Pushing the limits of low-resource morphological inflection. In *Proc. EMNLP*, Hong Kong.

Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2019. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3098–3112, Florence, Italy. Association for Computational Linguistics.

Çağrı Çöltekin. 2019. Cross-lingual morphological inflection with explicit alignment. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 71–79, Florence, Italy. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, et al. 2017. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. *arXiv preprint arXiv:1706.09031*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2015. Morphological inflection generation using character sequence to sequence learning. *arXiv preprint arXiv:1512.06110*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 634–643, San Diego, California. Association for Computational Linguistics.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain. Association for Computational Linguistics.

Ben Peters and André F. T. Martins. 2019. IT–IST at the SIGMORPHON 2019 shared task: Sparse two-headed models for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 50–56, Florence, Italy. Association for Computational Linguistics.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1357–1366, San Diego, California. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

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 Workshop on Computational Research in Phonetics, Phonology, and Morphology*.

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. *arXiv preprint arXiv:1905.06319*.