The NYU-CUBoulder Systems for SIGMORPHON 2020 Task 0 and Task 2

Assaf Singer
New York University
USA
as12152@nyu.edu

Katharina Kann
University of Colorado Boulder
USA
katharina.kann@colorado.edu

Abstract

We describe the NYU-CUBoulder systems for the SIGMORPHON 2020 Task 0 on typologically diverse morphological inflection and Task 2 on unsupervised morphological paradigm completion. The former consists of generating morphological inflections from a lemma and a set of morphosyntactic features describing the target form. The latter requires generating entire paradigms for a set of given lemmas from raw text alone. We model morphological inflection as a sequence-to-sequence problem, where the input is the sequence of the lemma's characters with morphological tags, and the output is the sequence of the inflected form's characters. First, we apply a transformer model to the task. Second, as inflected forms share most characters with the lemma, we further propose a pointer-generator transformer model to allow easy copying of input characters. Our best performing system for Task 0 is placed 6th out of 23 systems. We further use our inflection systems as subcomponents of approaches for Task 2. Our best performing system for Task 2 is the 2nd best out of 7 submissions.

1 Introduction

In morphologically rich languages, a word's surface form reflects syntactic and semantic properties that are expressed by the word. For example, most English nouns have both singular and plural forms (e.g., *robot*/*robots*, *process*/*processes*), which are known as the inflected forms of the noun. Some languages display little inflection. In contrast, others have many inflections per base form or lemma: a Polish verb has nearly 100 inflected forms (Janecki, 2000) and an Archi verb has around 1.5 million (Kibrik, 1998).

Morphological inflection is the task of, given an input word – a lemma – together with morphosyntactic features defining the target form, generating the indicated inflected form, cf. Figure 1. Morphological inflection is a useful tool for many natural language processing tasks (Seeker and Çetinoglu, 2015; Cotterell et al., 2016b), especially in morphologically rich languages where handling inflected forms can reduce data sparsity (Minkov et al., 2007).

Lemma	Features	Inflected form
hug	V;PST	hugged
seel	V;3;SG;PRS	seels

Figure 1: Morphological inflection examples in English. A lemma and features are mapped to an inflected form.

The SIGMORPHON 2020 Shared Task consists of three separate tasks. We participate in Task 0 on typologically diverse morphological inflection (Vylomova et al., 2020) and Task 2 on unsupervised morphological paradigm completion (Kann et al., 2020). **Task 0** consists of generating morphological inflections from a lemma and a set of morphosyntactic features describing the target form. For this task, we implement a pointer-generator transformer model, based on the vanilla transformer model (Vaswani et al., 2017) and the pointer-generator model (See et al., 2017). After adding a copy mechanism to the transformer, it produces a final probability distribution as a combination of generating elements from its output vocabulary and copying elements – characters in our case – from the input. As most inflected forms derive their characters from the source lemma, the use of a mechanism for copying characters directly from the lemma has proven to be effective for morphological inflection generation, especially in the low resource setting (Aharoni and Goldberg, 2017; Makarov et al., 2017).

For our submissions, we further increase the size of all training sets by performing multi-task train-

ing on morphological inflection and morphological reinflection, i.e., the task of generating inflected forms from forms *different from the lemma*. For languages with small training sets, we also perform hallucination pretraining (Anastasopoulos and Neubig, 2019), where we generate pseudo training instances for the task, based on suffixation and prefixation rules collected from the original dataset.

For **Task 2**, participants are given raw text and a source file with lemmas. The objective is to generate the complete paradigms for all lemmas. Our systems for this task consist of a combination of the official baseline system (Jin et al., 2020) and our systems for Task 0. The baseline system finds inflected forms in the text, decides on the number of inflected forms per lemma, and produces pseudo training files for morphological inflection. Our inflection model then learns from these and, subsequently, generates all missing forms.

2 Related Work

SIGMORPHON and CoNLL–SIGMORPHON shared tasks. In recent years, the SIGMORPHON and CoNLL–SIGMORPHON shared tasks have promoted research on computational morphology, with a strong focus on morphological inflection. Research related to those shared tasks includes Kann and Schütze (2016b), who used an LSTM (Hochreiter and Schmidhuber, 1997) sequence-to-sequence model with soft attention (Bahdanau et al., 2015) and achieved the best result in the SIGMORPHON 2016 shared task (Kann and Schütze, 2016a; Cotterell et al., 2016a). Due to the often monotonic alignment between input and output, Aharoni and Goldberg (2017) proposed a model with hard monotonic attention. Based on this, Makarov et al. (2017) implemented a neural state-transition system which also used hard monotonic attention and achieved the best results for Task 1 of the SIGMORPHON 2017 shared task. In 2018, the best results were achieved by a revised version of the neural transducer, trained with imitation learning (Makarov and Clematide, 2018). That model learned an alignment instead of maximizing the likelihood of gold action sequences given by a separate aligner.

Transformers. Transformers have produced state-of-the-art results on various tasks such as machine translation (Vaswani et al., 2017) language modeling (Al-Rfou et al., 2019), question answering (Devlin et al., 2019) and language understanding (Devlin et al., 2019). There has been very little work on transformers for morphological inflection, with, to the best of our knowledge, Erdmann et al. (2020) being the only published paper. However, the widespread success of transformers in NLP leads us to believe that a transformer model could perform well on morphological inflection.

Pointer-generators. In addition to the transformer, the architecture of our model is also inspired by See et al. (2017), who used a pointer-generator network for abstractive summarization. Their model could choose between generating a new element and copying an element from the input directly to the output. This copying of words from the source text via pointing (Vinyals et al., 2015), improved the handling of out-of-vocabulary words. Copy mechanisms have also been used for other tasks, including morphological inflection (Sharma et al., 2018). *Transformers* with copy mechanisms have been used for word-level tasks (Zhao et al., 2019), but, as far as we know, never before on the character level.

3 SIGMORPHON 2020 Shared Task

The SIGMORPHON 2020 Shared Task is composed of three tasks: Task 0 on typologically diverse morphological inflection (Vylomova et al., 2020), Task 1 on multilingual grapheme-to-phoneme conversion (Gorman et al., 2020), and Task 2 on unsupervised morphological paradigm completion (Kann et al., 2020). We submit systems to Tasks 0 and 2.

3.1 Task 0: Typologically Diverse Morphological Inflection

SIGMORPHON 2020 Task 0 focuses on morphological inflection in a set of typologically diverse languages. Different languages inflect differently, so it is not trivially clear that systems that work on some languages also perform well on others. For Task 0, systems need to generalize well to a large group of languages, including languages unseen during model development.

The task features 90 languages in total. 45 of them are development languages, coming from five families: Austronesian, Niger–Congo, Uralic, Oto-Manguean, and Indo-European. The remaining 45 are surprise languages, and many of those are from language families different from the development languages. Some languages have very small training sets, which makes them hard to model. For

those cases, the organizers recommend a family-based multilingual approach to exploit similarities between related languages. While this might be effective, we believe that using multitask training in combination with hallucination pretraining can give the model enough information to learn the task well, while staying true to the specific structure of each individual language.

3.2 Task 2: Unsupervised Morphological Paradigm Completion

Task 2 is a novel task, designed to encourage work on unsupervised methods for computational morphology. As morphological annotations are limited for many of the world's languages, the study of morphological generation in the low-resource setting is of great interest (Cotterell et al., 2018). However, a different way to tackle the problem is by creating systems that are able to use data without annotations.

For Task 2, a tokenized Bible in each language is given to the participants, along with a list of lemmas. Participants should then produce complete paradigms for each lemma. As slots in the paradigm are not labeled with gold data paradigm slot descriptions, an evaluation metric called best-match accuracy was designed for this task. First, this metric matches predicted paradigm slots with gold slots in the way which leads to the highest overall accuracy. It then evaluates the correctness of individual inflected forms.

4 Methods

In this section, we introduce our models for Tasks 0 and 2 and describe all approaches we use, such as multitask training, hallucination pretraining and ensembling. The code for our models is available online.[1]

4.1 Transformer

Our model is built on top of the transformer architecture (Vaswani et al., 2017). It consists of an encoder and a decoder, each composed of a stack of layers. Each encoder layer consists, in turn, of a self-attention layer, followed by a fully connected layer. Decoder layers contain an additional inter-attention layer between the two.

With inputs $(x_1, \cdots, x_T)$ being a lemma's characters followed by tags representing the morphosyntactic features of the target form, the encoder processes the input sequence and outputs hidden states $(h_1, \cdots, h_T)$. At generation step t, the decoder reads the previously generated sequence $(y_1, \cdots, y_{t-1})$ to produce states $(s_1, \cdots, s_{t-1})$. The last decoder state s_{t-1} is then passed through a linear layer followed by a softmax, to generate a probability distribution over the output vocabulary:

$$P_{\text{vocab}} = \text{softmax}(V s_{t-1} + b) \quad (1)$$

During training, the entire target sequence $\left(y_1, \cdots, y_{T_y}\right)$ is input to the decoder at once, along with a sequential mask to prevent positions from attending to subsequent positions.

4.2 Pointer-Generator Transformer

The pointer-generator transformer allows for both generating characters from a fixed vocabulary, as well as copying from the source sequence via pointing (Vinyals et al., 2015). This is managed by p_{gen} – the probability of generating as opposed to copying – which acts as a soft switch between the two actions. p_{gen} is computed by passing a concatenation of the decoder state s_t, the previously generated output y_{t-1}, and a context vector c_t through a linear layer, followed by the sigmoid function.

$$p_{\text{gen}} = \sigma(w[s_t; c_t; y_{t-1}] + b) \quad (2)$$

The context vector is computed as the weighted sum of the encoder hidden states

$$c_t = \sum_{i=1}^{T} a_i^t h_i \quad (3)$$

with attention weights $\left(a_1^t, \cdots, a_T^t\right)$. For each inflection example, let the extended vocabulary denote the union of the output vocabulary, and all characters appearing in the source lemma. We then use p_{gen}, P_{vocab} produced by the transformer, and the attention weights of the last decoder layer $\left(a_1^t, \cdots, a_T^t\right)$ to compute a distribution over the extended vocabulary:

$$P(c) = p_{\text{gen}} P_{\text{vocab}}(c) + (1 - p_{\text{gen}}) P_{\text{copy}}(c), \quad (4)$$

with

$$P_{\text{copy}}(c) = \sum_{i:x_i=c} a_i^t \quad (5)$$

The copy distribution $P_{\text{copy}}(c)$ for each character c is the sum of attention weights over all source positions where $x_i = c$. Note that if c is an out-of-vocabulary (OOV) character, then $P_{\text{vocab}}(c)$ is zero; similarly, if c does not appear in the source lemma,

[1] https://github.com/AssafSinger94/sigmorphon-2020-inflection

raw	grip	grips	V;SG;3;PRS
	grip	gripped	V;PST
generated	grips	grip	V;LEMMA
	grips	gripped	V;PST
	gripped	grip	V;LEMMA

Figure 2: English multitask training example (Task 0).

then $\sum_{i:x_i=c} a_i^t$ is zero. The ability to produce OOV characters is one of the primary advantages of pointer-generator models; by contrast models such as our vanilla transformer are restricted to their pre-set vocabulary.

4.3 Multitask Training

Some languages in Task 0 have small training sets, which makes them hard to model. In order to handle that, we perform multitask training, and, thereby, increase the amount of examples available for training.

Morphological reinflection. Morphological reinflection is a generalized version of the morphological inflection task, which consists of producing an inflected form for any given source form – i.e., not necessarily the lemma –, and target tag. For example:

$$\text{(hugging; V;PST)} \rightarrow \text{hugged.} \quad (6)$$

This is a more complex task, since a model needs to infer the underlying lemma of the source form in order to inflect it correctly to the desired form.

Many morphological inflection datasets contain lemmas that are converted to several inflected forms. Treating separate instances for the same source lemma as independent is missing an opportunity to utilize the connection between the different inflected forms. We approach this by converting our morphological inflection training set into one for morphological reinflection as described in the following.

From inflection to reinflection. Inflected forms of the same lemma are grouped together to sets of one or more (inflected form, morphological features) pairs. Then, for each set, we create new training instances by inflecting all forms to one another, as shown in Figure 2. We also let the model inflect forms back to the lemma by adding the lemma as one of the inflected forms, marked with the synthetically generated LEMMA tag. The new training set fully utilizes the connections between different forms in the paradigm, and, in that way, provides more training instances to our model.

Hyperparameter	Value
Embedding dimension	256
Encoder layers	4
Decoder layers	4
Encoder hidden dimension	1024
Decoder hidden dimension	1024
Attention heads	4

Table 1: The hyperparameters used in our inflection models for both Task 0 and Task 2.

4.4 Hallucination Pretraining

Another effective tool to improve training in the low-resource setting is data hallucination (Anastasopoulos and Neubig, 2019). Using hallucination, new pseudo-instances are generated for training, based on suffixation and prefixation rules collected from the original dataset. For languages with less than 1000 training instances, we pretrain our models on a hallucinated training set consisting of 10,000 instances, before training on the multitask training set.

4.5 Submissions and Ensembling Strategies

We submit 4 different systems for Task 0. NYU-CUBoulder-2 consists of one pointer-generator transformer model, and, for NYU-CUBoulder-4, we train one vanilla transformer. Those two are our simplest systems and can be seen as baselines for our other submissions.

Because of the effects of random initialization in non-convex objective functions, we further use ensembling in combination with both architectures: NYU-CUBoulder-1 is an ensemble of three pointer-generator transformers, and NYU-CUBoulder-3 is an ensemble of five pointer-generator transformers. The final decision is made by majority voting. In case of a tie, the answer is chosen randomly among the most frequent predictions. Models participating in the ensembles are from different epochs during the same training run.

As previously stated, all systems are trained on the augmented multitask training sets, and systems trained on languages with less than 1000 training instances were pretrained on the hallucinated datasets.

4.6 Task 2: Model description

Our systems for Task 2 consist of a combination of the official baseline system (Jin et al., 2020) and our inflection systems for Task 0. The system is given raw text and a source file with lemmas, and generates the complete paradigm of each lemma. The baseline system finds inflected forms in the text, decides on the number of inflected forms per lemma, and produces pseudo training files for morphological inflection. Any inflections that the system has not found in the raw text are given as test instances. Our inflection model then learns from the files and, subsequently, generates all missing forms. We use the pointer-generator and vanilla transformers as our inflection models.

For Task 2, we use ensembling for all submissions. NYU-CUBoulder-1 is an ensemble of six pointer-generator transformers, NYU-CUBoulder-2 is an ensemble of six vanilla transformers, and NYU-CUBoulder-3 is an ensemble of all twelve models. For all models in both tasks, we use the hyperparameters described in Table 1.

5 Experiments

5.1 Task 0

Data. The dataset for Task 0 covers 90 languages in total: 45 development languages and 45 surprise languages. For details on the official dataset please refer to Vylomova et al. (2020).

Baselines. This year, several baselines are provided for the task. The first system has also been used as a baseline in previous shared tasks on morphological reinflection (Cotterell et al., 2017, 2018). It is a non-neural system which first scans the dataset to extract suffix- or prefix-based lemma-to-form transformations. Then, based on the morphological tag at inference time, it applies the most frequent suitable transformation to an input lemma to yield the output form (Cotterell et al., 2017). The other two baselines are neural models. One is a transformer (Vaswani et al., 2017; Wu et al., 2020), and the second one is a hard-attention model (Wu and Cotterell, 2019), which enforces strict monotonicity and learns a latent alignment while learning to transduce. To account for the low-resource settings for some languages, the organizers also employ two additional methods: constructing a multilingual model trained for all languages belonging to each language family (Kann et al., 2017), and data augmentation using hallucination (Anastasopoulos and Neubig, 2019). Four model types are trained for each neural architecture: a plain model, a family-multilingual model, a data augmented model, and an augmented family-multilingual model. Overall, there are nine baseline systems for each language. We compare our models to an oracle baseline by choosing the best score over all baseline systems for each language.

	Sub-1	Sub-2	Sub-3	Sub-4	Base
Development Set					
Low	**88.71**	88.02	84.90	84.07	-
Other	90.46	90.63	90.20	**90.94**	-
All	**90.06**	90.02	88.96	89.34	-
Test Set					
Low	84.8	84.8	85.5	83.9	**89.77**
Other	89.7	89.8	89.8	90.2	**92.43**
All	88.6	88.7	88.8	88.8	**91.81**

Table 2: Macro-averaged results over all languages on the official development and test sets for Task 0. Low=languages with less than 1000 train instances, Other=all other languages, All=all languages.

Results. Our results for Task 0 are displayed in Table 2. All four systems produce relatively similar results. NYU-CUBoulder-3, our five-model ensemble, performs best overall with 88.8% accuracy on average. We further look at the results for low-resource ($<$ 1000 training examples) and high-resource ($>=$ 1000 training examples) languages separately. This way, we are able to see the advantage of the pointer-generator transformer in the low-resource setting, where all pointer-generator systems achieve an at least 0.9% higher accuracy than the vanilla transformer model. However, in the setting where training data is abundant, the effect of the copy mechanism vanishes, as NYU-CUBoulder-4 – our only vanilla transformer – achieved the best results for our high-resource languages.

5.2 Task 2

Data. For Task 2, a tokenized Bible in each language is given to the participants, along with a list of lemmas. Participants are required to construct the paradigms for all given lemmas.

The languages for Task 2 are again divided into development and test languages. Development languages are available for model development and hyperparameter tuning, but are not used during the final evaluation. The test languages are used for

System	Baseline 1		Baseline 2		Sub-1		Sub-2		Sub-3	
	Test Set									
	slots	macro	slots	macro	slots	macro	slots	macro	slots	macro
Basque	30	0.0006	27	0.0006	30	0.0005	30	0.0005	30	**0.0007**
Bulgarian	35	0.283	34	**0.3169**	35	0.2769	35	0.2894	35	0.2789
English	4	0.656	4	**0.662**	4	0.502	4	0.528	4	0.512
Finnish	21	0.0533	21	**0.055**	21	0.0536	21	0.0547	21	0.0535
German	9	0.2835	9	**0.29**	9	0.273	9	0.2735	9	0.2735
Kannada	172	0.1549	172	**0.1512**	172	0.111	172	0.1116	172	0.111
Navajo	3	0.0323	3	**0.0327**	3	0.004	3	0.0043	3	0.0043
Spanish	29	0.2296	29	**0.2367**	29	0.2039	29	0.2056	29	0.203
Turkish	104	0.1421	104	**0.1553**	104	0.1488	104	0.1539	104	0.1513
All		0.2039		**0.2112**		0.1749		0.1802		0.1765

Table 3: Results for all test languages on the official test sets for Task 2.

evaluation only, and do not have development sets. The development languages are: Maltese, Persian, Portuguese, Russian, Swedish. The test languages are: Basque, Bulgarian, English, Finnish, German, Kannada, Navajo, Spanish and Turkish.

Baselines. The baseline system for the task is composed of four components, eventually producing morphological paradigms (Jin et al., 2020). The first three modules perform edit tree (Chrupala, 2020) retrieval, additional lemma retrieval from the corpus, and paradigm size discovery, using distributional information. After the first three steps, pseudo training and test files for morphological inflection are produced. Finally, the non-neural Task 0 baseline system (Cotterell et al., 2017) or the neural transducer by Makarov and Clematide (2018) are used to create missing inflected forms.

Results. Systems for Task 2 are evaluated using macro-averaged best-match accuracy (Jin et al., 2020). Results are shown in in Table 3. All three systems produce relatively similar results. NYU-CUBoulder-2, our vanilla transformer ensemble, performed slightly better overall with an average best-match accuracy of 18.02%. Since our system is close to the baseline models, it performs similarly, achieving slightly worse results. For Basque, our all-round ensemble NYU-CUBoulder-2 outperformed both baselines with a best-match accuracy of 00.07%, achieving the highest result in the shared task.

5.3 Low-resource Setting

As most inflected forms derive their characters from the source lemma, the use of a mechanism for copying characters directly from the lemma has proven to be effective for morphological inflection generation, especially in the low-resource setting (Aharoni and Goldberg, 2017; Makarov et al., 2017). As all Task 0 datasets are fairly large, we further design a low-resource experiment to investigate the effectiveness of our model.

Data. We simulate a low-resource setting by sampling 100 instances from all languages that we already consider low-resource, i.e., all languages with less than 1000 training instances. We then keep their development and test sets unchanged. Overall, we perform this experiment on 21 languages.

Experimental setup. We train a pointer-generator transformer and a vanilla transformer on the modified datasets to examine the effects of the copy mechanism. We keep the hyperparameters unchanged, i.e., they are as mentioned in Table 1. We use a majority-vote ensemble consisting of 5 individual models for each architecture.

Baseline. We additionally train the neural transducer by Makarov and Clematide (2018), which has achieved the best results for the 2018 shared task in the low-resource setting (Cotterell et al.,

System	Trm	Trm-PG	Baseline
All	63.06	67.61	**70.06**

Table 4: Results on the official development data for our low-resource experiment. Trm=Vanilla transformer, Trm-PG=Pointer-generator transformer, Baseline=neural transducer by Makarov and Clematide (2018).

Model:	1	2	3	4	5
Copy	✓	✓			✓
Multitask Train	✓		✓		✓
Hallucination	✓	✓	✓	✓	

Table 5: System components for the ablation study for Task 0. Each model is a transformer which contains a combination of the following components: copy mechanism, multitask training and hallucination pretraining.

Model:	1	2	3	4	5
Development Set					
Low	88.20	**90.00**	87.52	89.84	86.35
Other	90.63	**92.66**	90.93	92.60	90.63
All	90.02	**92.04**	90.13	91.96	89.63

Table 6: Ablation study for Task 0; development set results, averaged over all languages. Low=languages with less than 1000 train instances, Other=all other languages, All=all languages.

2018). The neural transducer uses hard monotonic attention (Aharoni and Goldberg, 2017) and transduces the lemma into the inflected form by a sequence of explicit edit operations. It is trained with an imitation learning method (Makarov and Clematide, 2018). We use this model as a reference for the state of the art in the low-resource setting.

Results. As seen in Table 4, for the low-resource dataset, the pointer-generator transformer clearly outperforms the vanilla transformer by an average accuracy of 4.46%. For some languages, such as Chichicapan Zapotec, the difference is up to 14%. While the neural transducer achieves a higher accuracy, our model performs only 2.45% worse than this state-of-the-art model.[2] We are also able to observe the use of the copy mechanism for copying of OOV characters in the test sets of some languages.

6 Ablation Studies

Our systems use three components on top of the vanilla transformer: a copy mechanism, multitask training and hallucination pretraining. We further perform an ablation study to measure the contribution of each component to the overall system performance. For this, we additionally train five different systems with different combinations of components. A description of which component is used in which system for this ablation study is shown in Table 5.

6.1 Results

Copy mechanism. Comparing models 2 and 4, which are both trained on the original dataset, pretrained with hallucination and differ only by the use of the copy mechanism, we are able to see that adding this component slightly improves performance by $0.06-0.16\%$. When comparing models 1 and 3, the copy mechanism decreases performance slightly by 0.3% for the high-resource languages and 0.11% overall, but increases performance for low-resource languages by 0.68%.

Multitask training. Unlike the copy mechanism, multitask training actually consistently decreases the performance of the models. Looking at models 1 and 2, training the pointer-generator transformer on the multitask dataset decreases accuracy by $1.8 - 2.03\%$ for all three language groups. The same happens for the vanilla transformer with an accuracy decrease of $1.67 - 2.32\%$. A possible explanation are the relatively large training sets provided for the shared task, as this method is more suitable for the low-resource setting.

Hallucination pretraining. In order to examine the effect of hallucination pretraining on our submitted models, we now compare the pointer-generator transformers trained on the multitask data with and without hallucination pretraining (models 1 and 5). Hallucination pretraining shows to be helpful: it increases the accuracy on low-resource languages by 1.85%. The performance on the high-resource languages is necessarily the same, as only models for low-resource languages are actually pretrained.

7 Conclusion

We presented the NYU-CUBoulder submissions for SIGMORPHON 2020 Task 0 and Task 2.

We developed morphological inflection models, based on a transformer and a new model for the task, a pointer-generator transformer, which is a transformer-analogue of a pointer-generator model. For Task 0, we further added multitask training and hallucination pretraining. For Task 2, we combined our inflection models with additional components from the provided baseline to obtain a fully functional system for unsupervised morphological paradigm completion.

We performed an ablation study to examine the

[2] We could probably obtain better results with appropriate hyperparameter tuning.

effects of all components of our inflection system. Finally, we designed a low-resource experiment to show that using the copy mechanism on top of the vanilla transformer is beneficial if training sets are small, and achieved results close to a state-of-the-art model for low-resource morphological inflection.

Acknowledgments

We would like to thank the organizers of SIGMORPHON 2020 Task 0 and Task 2.

References

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 2004–2015.

Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2019. Character-level language modeling with deeper self-attention. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019.*, pages 3159–3166.

Antonios Anastasopoulos and Graham Neubig. 2019. Pushing the limits of low-resource morphological inflection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 984–996. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Grzegorz Chrupala. 2020. Towards a machine-learning architecture for lexical functional grammar parsing.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, S. J. Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The conll-sigmorphon 2018 shared task: Universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection, Brussels, October 31 - November 1, 2018*, pages 1–27. 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, and Mans Hulden. 2017. Conll-sigmorphon 2017 shared task: Universal morphological reinflection in 52 languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection, Vancouver, BC, Canada, August 3-4, 2017*, pages 1–30. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016a. The SIGMORPHON 2016 shared task - morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, Berlin, Germany, August 11, 2016*, pages 10–22. Association for Computational Linguistics.

Ryan Cotterell, Hinrich Schütze, and Jason Eisner. 2016b. Morphological smoothing and extrapolation of word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Alexander Erdmann, Micha Elsner, Shijie Wu, Ryan Cotterell, and Nizar Habash. 2020. The paradigm discovery problem. *CoRR*, abs/2005.01630.

Kyle Gorman, Lucas F.E. Ashby, Aaron Goyzueta, Arya D. McCarthy, Shijie Wu, and Daniel You. 2020. The sigmorphon 2020 shared task on multilingual grapheme-to-phoneme conversion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Klara Janecki. 2000. 300 polish verbs. Barron's Educational Series.

Huiming Jin, Liwei Cai, Yihui Peng, Chen Xia, Arya D. McCarthy, and Katharina Kann. 2020. Unsupervised morphological paradigm completion. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1993–2003, Vancouver, Canada. Association for Computational Linguistics.

Katharina Kann, Arya D. McCarthy, Garrett Nicolai, and Mans Hulden. 2020. The SIGMORPHON 2020 shared task on unsupervised morphological paradigm completion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics.

Katharina Kann and Hinrich Schütze. 2016a. MED: the LMU system for the SIGMORPHON 2016 shared task on morphological reinflection. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, Berlin, Germany, August 11, 2016*, pages 62–70.

Katharina Kann and Hinrich Schütze. 2016b. Single-model encoder-decoder with explicit morphological representation for reinflection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*.

Aleksandr E. Kibrik. 1998. The handbook of morphology. In *Andrew Spencer and Arnold M. Zwicky, editors*, pages 455–476. Oxford: Blackwell Publishers.

Peter Makarov and Simon Clematide. 2018. UZH at conll-sigmorphon 2018 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection, Brussels, October 31 - November 1, 2018*, pages 69–75. Association for Computational Linguistics.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. *CoRR*, abs/1707.01355.

Einat Minkov, Kristina Toutanova, and Hisami Suzuki. 2007. Generating complex morphology for machine translation. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083.

Wolfgang Seeker and Özlem Çetinoglu. 2015. A graph-based lattice dependency parser for joint morphological segmentation and syntactic analysis. *Trans. Assoc. Comput. Linguistics*, 3:359–373.

Abhishek Sharma, Ganesh Katrapati, and Dipti Misra Sharma. 2018. IIT(BHU)-IIITH at conll-sigmorphon 2018 shared task on universal morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection, Brussels, October 31 - November 1, 2018*, pages 105–111.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2773–2781.

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

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1530–1537. Association for Computational Linguistics.

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

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 156–165.