Generalizing Morphological Inflection Systems to Unseen Lemmas

Changbing Yang* **Ruixin (Ray) Yang*** **Garrett Nicolai** **Miikka Silfverberg**
University of British Columbia
`first.last@ubc.ca`

Abstract

This paper presents experiments on morphological inflection using data from the SIGMORPHON-UniMorph 2022 Shared Task 0: Generalization and Typologically Diverse Morphological Inflection. We present a transformer inflection system, which enriches the standard transformer architecture with reverse positional encoding and type embeddings. We further apply data hallucination and lemma copying to augment training data. We train models using a two-stage procedure: (1) We first train on the augmented training data using standard backpropagation and teacher forcing. (2) We then continue training with a variant of the scheduled sampling algorithm dubbed student forcing. Our system delivers competitive performance under the small and large data conditions on the shared task datasets.

1 Introduction

This paper presents experiments on morphological inflection using data from the SIGMORPHON-UniMorph Shared Task 0: Generalization and Typologically Diverse Morphological Inflection (Kodner et al., 2022).[1] Our system focuses on typologically diverse inflection generation, that is, the task of inflecting a lemma in a given form, which is specified by a morphosyntactic description (MSD). As an example, consider inflecting the English verb lemma *walk* in the past tense according to the MSD `VERB+PAST`, thereby generating the inflected form *walked*. The shared task investigates two data conditions: Under the *small data condition*, up to 700 training examples are provided. Under the *large data condition*, up to 7000 training examples are provided. Our system beats the official neural shared task baseline by more than 8%-points under both the small and large data conditions.

We apply *transformer models* (Vaswani et al., 2017b) to the inflection task. The model is trained to translate an input sequence consisting of lemma characters and an MSD, like:

```
w, a, l, k, +VERB, +PAST
```

into the inflected output sequence:

```
w, a, l, k, e, d
```

General purpose transformers were originally developed for machine translation, but they also deliver strong performance on morphology tasks (Wu et al., 2021). Nevertheless, we observe that the vanilla transformer architecture is not ideally suited for inflection: In contrast to machine translation, many inflectional phenomena are strongly positionally dependent, which is something that the vanilla transformer architecture does not adequately model. For example, phonological alternations often happen at affix boundaries and these typically occur either at the start or end of word forms. Whereas the positional encoding in the transformer architecture allows for uniquely conditioning on relative positions with regard to the start of the string, the same is not true for positions at the end of the input string. We, therefore, augment our transformers with *reverse positional encoding*, presented in Section 3.1, which allow the model to condition directly on the end of the input string.

In previous iterations of the SIGMORPHON inflection shared task (Pimentel et al., 2021; Vylomova et al., 2020; McCarthy et al., 2019; Cotterell et al., 2018, 2017, 2016), so called *lemma overlap*, where identical lemmas occur both in the training and test set, has caused inflated performance, resulting in near perfect inflection accuracy for many languages. Liu and Hulden (2022) and Goldman et al. (2022) show that more challenging data splits with low lemma overlap can cause significant reduction in inflection performance. The data in this

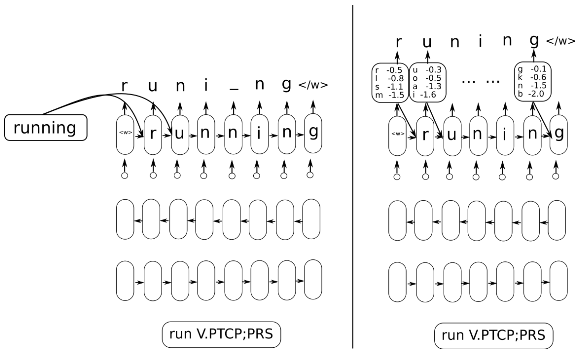
*The first two authors contributed equally.

[1]Note, our system is not an official shared task submission because we submitted our final results after the shared task deadline.

year's inflection task demonstrate varying lemma overlap, ranging from < 1% for Slovak under the small data condition to 100% for Hebrew under the large data condition but centering on lower overlap (see Appendix A for details). Accordingly, we decided to investigate different mechanisms which we hypothesized would improve generalization to unseen lemmas in the test set.

Data augmentation is a commonly used technique, which improves generalization in many NLP tasks. Here the gold standard training data is augmented with synthetic examples. Back-translation introduced for machine translation is perhaps the best known method (Sennrich et al., 2016), but has not been very successful in morphology tasks (Liu and Hulden, 2021). We instead use the *data hallucination* approach by Anastasopoulos and Neubig (2019), which synthesizes new training examples from existing gold standard training examples by identifying a (possibly discontinuous) word stem and replacing this with a random character sequence. In addition to data hallucination, we experiment with another data augmentation technique: *lemma copying* (Liu and Hulden, 2022), where the model is trained to copy input lemmas from the test set in order to adapt the model more closely to the test data. In our experiments, this method ultimately delivers better performance than data hallucination.

As a further attempt to improve generalization, we experiment with modifications of the standard *teacher forced* training procedure of inflection models. When applying teacher forcing during training, the model is allowed to rely on gold standard history for time steps 1 up to t, when predicting output at time step $t + 1$. This speeds up convergence considerably but can also result in sub-optimal performance due to so-called *exposure bias* (Wiseman and Rush, 2016), which is caused by a mismatch when conditioning on gold standard history during training and predicted history during test time. We take an alternative approach called *student forcing* (Nicolai and Silfverberg, 2020), which is an application of *scheduled sampling* (Bengio et al., 2015) for morphology tasks. Here model-predicted output history is substituted for the gold standard history for a subset of training examples in order to counteract exposure bias while simultaneously maintaining efficient training (see Figure 1). According to Nicolai and Silfverberg (2020), student forcing can improve inflection performance under



[Illustration from Nicolai and Silfverberg (2020)]

Figure 1: Teacher forcing (left) and student forcing (right); some connections have been left out to reduce clutter.

low-resource conditions. Our experiments show that student forcing can deliver small improvements for some languages but does not outperform data hallucination. However, the techniques seem to be complementary; their combination provides improvements over plain data augmentation.

In summary, our main contributions are as follow:

1. We enrich the transformer architecture with reverse positional encoding in order to support the inflection task.

2. We investigate data hallucination and lemma copying as ways to prompt better generalization to lemmas missing from the training set.

3. We apply student forcing to counter exposure bias in inflection.

2 Related Work

Wu et al. (2021) present a systematic investigation of applying the transformer model to morphology tasks. They propose two changes to the general transformer architecture introduced by Vaswani et al. (2017b): (1) type embeddings, which are used to distinguish between input characters and morphosyntactic tags and (2) restricting positional encoding to the input characters, while encoding morphosyntactic tags in a position-agnostic manner. Another modification to the transformer architecture, which can improve performance on morphology tasks, is to add a so-called monotonicity loss (Rios et al., 2021). This can bias the transformer toward near-monotonic alignment between the input and output sequence, which is often the case in inflection.

We use data augmentation to improve generalization to unseen lemmas. This has become a standard technique in low-resource inflection in recent years. A common approach is to generate synthetic examples by first identifying word stems in gold standard examples and then replacing the stems with random character sequences (Anastasopoulos and Neubig, 2019; Silfverberg et al., 2017). Liu and Hulden (2022) introduce a more refined method to hallucinate synthetic stems, which aims to honor the phonology of the target language by generating sequences of random syllables rather than random characters. Kann and Schütze (2017) show that a simpler data augmentation method, where random strings or unlabeled word forms are copied from the input to the output, can also be effective. Liu and Hulden (2022) apply this approach to copying lemmas in the development and test set and show that this can lead to substantial gains in inflection accuracy. We apply their technique in Section 3.4. Other approaches to data augmentation in morphological inflection include: reframing the task as reinflection and generating reinflection examples from the existing inflection training data (Liu and Hulden, 2020), as well as generating new training examples using back-translation (Liu and Hulden, 2021), and self-training (Yu et al., 2020).

In addition to data augmentation, we also experiment with student forcing to improve generalization. As mentioned above, this is an application of scheduled sampling. Bengio et al. (2015) explore scheduled sampling for various sequence generation tasks (image captioning, constituency parsing and speech recognition). This is a curriculum learning approach (Bengio et al., 2009), where the model is gradually exposed to more of its own prediction errors during training, thereby counteracting exposure bias. The student forcing approach presented by Nicolai and Silfverberg (2020) is a slight simplification of this approach. Essentially, student forcing uses a fixed amount of model-predicted contexts throughout training instead of a curriculum approach.

3 Methods

In this section, we describe our contributions to the inflection task, before moving on to our experiments in subsequent sections.

3.1 Reverse Positional Encoding

The vanilla Transformer architecture, which serves as the basis for our system, accounts for the order of input and output tokens by pairing each token with a sinusoidal positional encoding (Vaswani et al., 2017a). This positional encoding captures relative distance from the *start* of the string, meaning that it is a *forward* positional encoding. In inflection tasks, it is, however, vital to encode not only distance from the start of the input string, but also distance to the *end* of the string.

For example, in English, the plural form of nouns ending in a strident like *s* is formed by appending an affix *-es* to the end of the noun (e.g. *class* $\rightarrow$ *class+es*) instead of the regular plural suffix *-s*. The alternation *s* $\rightarrow$ *es* always occurs at the penultimate position of the inflected form, which means that it is important to allow the model to directly refer to positions at the end of the strings. Because word length differs, this information is difficult to infer from a purely forward positional encoding.

We augment the vanilla transformer model in the Fairseq toolkit (Ott et al., 2019) with reverse positional encoding: Let $f_1, ..., f_n$ be the k-dimensional forward sinusoidal positional encoding vectors for a string of length n. We introduce k-dimensional reverse positional encoding vectors $b_1, ..., b_n$, where $r_i = f_{n-i+1}$. Our final positional encoding vectors are given by the $2k$-dimensional concatenation $[f_i; b_i]$. Following Wu et al. (2021), we only use positional encoding vectors for characters in the input lemma. For morphosyntactic tags, we instead use a special NULL vector. See Figure 2 for a representation of the reverse positional encoding.

3.2 Type Embeddings

Given an example like *bus*+NOUN+PL $\rightarrow$ *buses*, the input sequences to our inflection model consist of two token-types: lemma-characters like *b, u* and *s* and morphosyntactic tags like +NOUN and +PL. Following Wu et al. (2021), we use type embedding vectors e_{LEM} and e_{MSD} to distinguish between these token-types. The type vectors have the same dimensionality as the input embeddings. We sum them with token embedding vectors to compute input token representations. The vectors e_{LEM} and e_{MSD} are randomly initialized and are trained jointly with the rest of the inflection model. See Figure 2 for an illustration of type embeddings.

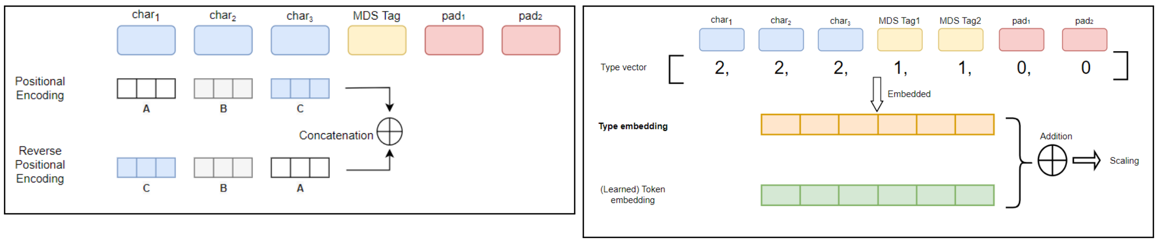
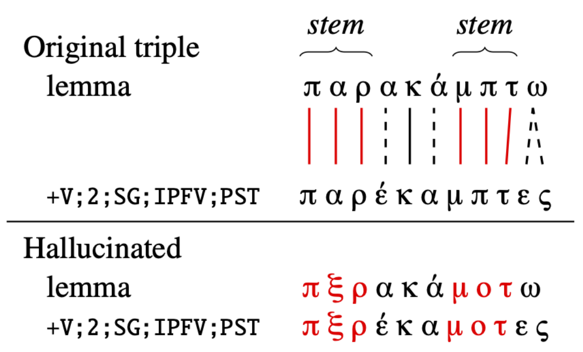


Figure 2: Illustration of reverse positional encoding and type embeddings. The left figure shows the encoding of source character positions from the backward pass, concatenated with the forward positional encoding. The right figure displays a type embedding built from an integer-encoded type vector that distinguishes the three possible types of an input token. The type embedding is then summed with the original token embedding and multiplied by a scaling factor.



[Illustration from Anastasopoulos and Neubig (2019)]

Figure 3: Illustration of the data hallucination method. Noise is introduced into the existing training examples by replacing the longest common subsequence of input and output forms with random character strings.

3.3 Data Hallucination

Under low-data conditions, encoder-decoder models are often strongly influenced by the target language model. Common character sequences which appear in the training data are more likely to be produced, even at the expense of ignoring the input example. In order to address this label bias, we augment the training data with hallucinated examples. We employ the approach proposed by Anastasopoulos and Neubig (2019). This method introduces noise into the existing training examples by replacing the longest common subsequence of input and output forms with random character strings, as shown in Figure 3.

Although the problem is more prevalent under low-data conditions, we experiment with adding synthetic examples to the original dataset under both the small and large data condition. Preliminary development experiments motivate the number of hallucinated forms. Accordingly, we use 7,000 synthetic examples for the small data set and 1,400 examples for the large training set.

3.4 Lemma Copying

The data hallucination method introduced by Anastasopoulos and Neubig (2019) can sometimes create invalid examples due to phonological alternations as noted by Samir and Silfverberg (2022). For example, given the English inflection example *like+VERB+PAST* $\rightarrow$ *liked*, their approach will first identify the longest common subsequence of the lemma and word form, that is, *like* and will then replace this with a random character sequence, for example *xyz*. This results in a synthetic example *xyz+VERB+PAST* $\rightarrow$ *xyzd*. Now, this example is erroneous since *-d* occurs as the English past tense marker for regular verbs only when the stem ends in *e*, which the syntetic stem *xyz* does not.

In order to avoid introducing errors during augmentation, we experiment with an alternative approach to data augmentation: so-called lemma copying, first presented by Liu and Hulden (2022). We augment the training set with artificial examples where a lemma is copied verbatim, e.g. *like*+COPY $\rightarrow$ *like*. Here we use the special +COPY tag to indicate copying. We collect lemmas for the copy examples from the input forms in the test set. Therefore, lemma copying can be seen as a domain adaptation technique, where we adapt the inflection model to the specific test input forms.

At a first glance, lemma copying might seem like an artificial technique, which will only be useful in a shared task setting where we have a fixed test set. However, even in real-world scenarios, we will often run the model on a fixed dataset of inputs.[2] It is, therefore, possible to either retrain the model on a combination of the original training data and test input forms, or fine-tune the model on lemma

[2]For example, we might want to inflect a set of baseforms from a dictionary.

copying.[3] It is also important to note that lemma copying does not use any additional labeled data for training the system. Neither does is make use of any additional unlabeled data, which would be unavailable at inference-time.

3.5 Student Forcing

Sequence-to-sequence architectures are very dependent on the context of generated items—it is their greatest strength, but can also lead to disjunctions between training and testing settings.

In very low data setups, exposure bias can overfit to the training data, as it observes a very small set of contexts. Although data hallucination has been shown to counter overfitting in such scenarios, we additionally adopt the student forcing approach described by Nicolai and Silfverberg (2020).

For a small number of instances (a tunable hyperparameter, *student-forced percentage* [SF-%], most effective between 10 and 30%), contextual cues from the target are replaced with hypotheses generated by the model. Hypotheses are typically generated via the standard inference method (in this case, a beam search with beam width 5). We explore several alternative methods to further allow the model to take advantage of the prediction space, including sampling from items that reach a probability threshold, a count threshold, and using multiple diverse beam groups. Development results suggested that sampling from the top 2 candidates yielded the best results, and is used for all experiments describing student forcing for the remainder of this paper.

Since hallucinated data makes up a significant portion of the training data (90% under the small data condition, and 17% under the large data condition), we anticipate the possibility that the model overfits to hallucinated data. In an attempt to counter overfitting, we apply student-forcing in a fine-tuning step after the initial data-augmented models have been trained.

4 Experiments and Results

Here, we describe our experiments on small and large training sets. Under both data conditions, we train models using the following procedure: We first augment the training data using data hallucination or lemma copying. We then train the model on the augmented data for a maximum of 20,000 steps without teacher forcing. We then identify the best checkpoint model based on development set accuracy and continue training this model with student forcing for an additional 1000 steps. When applying lemma copying, we have to train separate models for the development and test set: one model which augments the training set with lemmas from the development set and another one which augments with test lemmas. We first tune hyperparameters on the development set and then use this hyperparameter configuration when training the final model for the test set. Crucially, this allows us to avoid augmenting the training data both with development and test lemmas in order to not use extra data for tuning model parameters.

4.1 Original Data for Inflection Generation

Data across 33 languages are included in our experiments. We follow the training, development, and testing splits provided by the task organizers. Twenty of the languages contain two training conditions: small and large. Small training data range from 70 to 700 instances, where an instance is composed of a lemma, an MSD, and an inflected form. Most languages have 700 training instances, but Chukchi (ckt), Upper Sorbian (hsb), Kholosi (hsi), and Ket (ket) represent an even lower-resource condition. In the large training data condition, each language has 7,000 training instances. Generally, development splits contain approximately 1,000 instances, and test splits contain 2,000.

4.2 Model Architecture

We conduct our experiments with a modified version of Fairseq's (Ott et al., 2019) implementation of transformers (Vaswani et al., 2017b). The transformer architecture is enriched with reverse positional encoding and type embeddings, as we illustrated in Sections 3.1 and 3.2. We train our models with 4 layers in the encoder and decoder, each containing 4 attention heads. The embedding size is 256 and the hidden layer size is 1024. These hyperparameter settings roughly correspond to the values used by Wu et al. (2021) for character-level tasks.

We use the Adam optimizer with an initial learning rate of 0.001, and batch size 400. Prediction is performed with the best checkpoint model, according to the development accuracy, using a beam of width 5. All models are trained for a maximum of 20,000 updates. Fine-tuning then proceeds for a maximum of 1000 additional updates. Again, we

[3] In the current submission, we only investigate the retraining approach.

choose the best model as determined by development accuracy.

4.3 Main Results

Experiment	Small	Large
ST BASELINE	47.63	62.39
OUR BASELINE	47.93	69.57
HALL	53.83	69.19
COPY	56.64	70.66
COPY+SF	**57.23**	**71.26**
COPY+HALL	55.27	70.43

Table 1: Results on the test data under both small and large data conditions. ST BASELINE refers to the official neural shared task baseline and "Our Baseline" to our baseline transformer with reverse positional encoding and type embeddings. SF refers to student forcing, HALL to data hallucination and COPY to lemma copying.

We use micro averaged full-form accuracy to evaluate our predictions on development and test splits, including results both under the small and large data condition.[4] Average results across all languages are shown in Table 1.[5] See Kodner et al. (2022) for detailed results. The best results (Copy+SF) represent our official shared task submission.

Across both data conditions, our models outperform the official shared task neural baseline. Our modified Fairseq models with reverse positional encoding and type embeddings but without data augmentation (OUR BASELINE) perform slightly better than the official shared task baseline under the small training data condition, while on the large training set our modifications to the transformer architecture contribute a substantial improvement of around 7%-points.

Results from data hallucination HALL are mixed. Under the low data condition, it delivers a clear improvement of 5.90%-points over OUR BASELINE on the test set, but under the large data condition, it results in a small drop of 0.38%-points in inflection accuracy. In contrast, lemma copying delivers consistent improvements over OUR BASELINE under all data conditions. Under the small data condition, the COPY system delivers a substantial 8.71%-point improvement and a smaller improvement of 1.09%-points under the large data condition, outperforming HALL under both conditions. A combination of the data augmentation techniques COPY+HALL does not deliver improvements over plain lemma copying but outperforms HALL. In general, data augmentation is always more helpful under the low data condition.

Student forcing (COPY+SF) further boosts the performance of the COPY system for several languages, resulting in a 0.5%-points gain under both data conditions. Some languages show only modest improvement, such as Hebrew increasing from 34.6% to 35.2%, or even small decreases - Braj decreases from 56.1% to 56.0%. However, other improvements are much more noteworthy - Arabic increases from 43% to 47.9%, and Pomak from 44.2% to 46.0%. The trends are similar under the large data condition, although fewer languages are affected.

We take a closer look at the types of errors that are corrected by the COPY+SF model when compared to COPY. Concentrating on Evenki, we notice that the corrections made by student forcing are generally small - typically, the addition or removal of a single letter. For example, the 3rd person singular possessive form of *atirkanma* should be *atirkanman*. While the model prior to fine-tuning simply copies the lemma, COPY+SF corrects the error. Likewise, the 3rd person dative possessive form of *nadiśi* is predicted as *nadiśidun*, which is then corrected by student forcing to *nadiśidu:n*.

5 Discussion

The most prominent trend in our experiments is that lemma copying delivers sizable improvements in accuracy, particularly under the small data condition. It is also noteworthy that models trained on small training data using data augmentation(either hallucinated data or copied data) outperform models trained on large training data without data augmentation. Based on these results, it is clear that data augmentation is a crucial technique in low-resource inflection, delivering substantial improvements which parallel improvements from a significant additional annotation effort. This might allow researchers to kick-start development of morphology resources for low-resource languages using very little annotated data. Performance also seems to improve even under higher data conditions when lemma overlap in the training data and test data is small.

Student forcing delivers small improvements at best and is often harmful when combined with data augmentation. We do not have a good explanation

[4]This corresponds to the official evaluation metric of the SIGMORPHON 2022 inflection shared task.

[5]See Appendix B for results on the development set.

for this phenomenon at the current time. Based on our experimental results, we can conclude that data augmentation is a far more influential method for countering data sparsity.

It is interesting to see that our base inflector, trained without student forcing or data augmentation, outperforms the shared task baseline. Given that the baseline system is a character-level transformer (Wu et al., 2021), this might be attributable to our architectural innovation, namely reverse positional encoding. However, another difference between our system and the shared task baseline is that the baseline is a multilingual system, whereas our system is monolingual. Further investigation is required to tease apart these effects.

6 Conclusion

In this work, we advance the generation performance of inflectional forms with a joint effort including reverse positional encoding, data hallucination, copying lemmas, and student forcing. We improve the prediction accuracy by 9.6% and 8.6% above the official neural shared task baseline on the small and large test set respectively.

According to our results, the joint effect of reverse positional encoding, lemma copying, and student forcing results in the best performance. We investigate two data augmentation strategies: The effect of data augmentation is more evident when less annotated data is available for training.

Due to time constraints, many observed phenomena are still ripe for interpretation, including the role that sampling has in a space populated by artificial examples. Our findings suggest that not only is data hallucination beneficial for low-resource morphological inflection, but that it is a necessary step in the inflectional pipeline. That said, there is still room to improve. Even in the more challenging (and more realistic) setting present in this task, several languages are close to solved for inflection, but many still have significant room for improvement. We anticipate more focused investigations into the reasons why these languages remain so difficult for transformer models, even as the state of the art approaches new heights.

Acknowledgements

We want to thank Farhan Samir for useful discussions regarding scheduled sampling. We also want to thank the organizers of the shared task. This research was supported by funding from the National Endowment for the Humanities (Documenting Endangered Languages Fellowship) and the Social Sciences and Humanities Research Council of Canada (Grant 430-2020-00793). Any views/findings/conclusions expressed in this publication do not necessarily reflect those of SSHRC.

References

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*, pages 984–996.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems*, 28.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sabrina 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*, pages 1–27.

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*, pages 1–30.

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

Omer Goldman, David Guriel, and Reut Tsarfaty. 2022. (un)solving morphological inflection: Lemma overlap artificially inflates models' performance. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 864–870.

Katharina Kann and Hinrich Schütze. 2017. Unlabeled data for morphological generation with character-based sequence-to-sequence models. In *Proceedings* *of the First Workshop on Subword and Character Level Models in NLP*, pages 76–81.

Jordan Kodner, Salam Khalifa, Khuyagbaatar Batsuren, Hossep Dolatian, Ryan Cotterell, Faruk Akkuş, Antonios Anastasopoulos, Taras Andrushko, Aryaman Arora, Nona Atanelov, Gábor Bella, Elena Budianskaya, Yustinus Ghanggo Ate, Omer Goldman, Simon Guriel, Silvia Guriel-Agiashvili, Jan Hajič, Jan Hric, Ritvan Karahodja, Witold Kieraś, Andrew Krizhanovsky, Natalia Krizhanovsky, Igor Marchenko, Magdalena Markowska, Polina Mashkovtseva, Maria Nepomniashchaya, Daria Rodionova, Elizabeth Salesky, Karina Sheifer, Alexandra Serova, Anastasia Yemelina, Jeremiah Young, and Ekaterina Vylomova. 2022. SIGMORPHON-UniMorph 2022 Shared Task 0: Generalization and Typologically Diverse Morphological Inflection. In *Proceedings of the SIGMORPHON 2022 Shared Task: Morphological Inflection*.

Ling Liu and Mans Hulden. 2020. Leveraging principal parts for morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 153–161.

Ling Liu and Mans Hulden. 2021. Backtranslation in neural morphological inflection. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pages 81–88.

Ling Liu and Mans Hulden. 2022. Can a transformer pass the wug test? tuning copying bias in neural morphological inflection models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 739–749.

Arya D. McCarthy, Ekaterina Vylomova, Shijie Wu, Chaitanya Malaviya, Lawrence Wolf-Sonkin, Garrett Nicolai, Christo Kirov, Miikka Silfverberg, Sabrina J. Mielke, Jeffrey Heinz, Ryan Cotterell, and Mans Hulden. 2019. The SIGMORPHON 2019 shared task: Morphological analysis in context and cross-lingual transfer for inflection. In *Proceedings of the 16th Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–244.

Garrett Nicolai and Miikka Silfverberg. 2020. Noise isn't always negative: Countering exposure bias in sequence-to-sequence inflection models. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2837–2846.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53.

Tiago Pimentel, Maria Ryskina, Sabrina J. Mielke, Shijie Wu, Eleanor Chodroff, Brian Leonard, Garrett Nicolai, Yustinus Ghanggo Ate, Salam Khalifa, Nizar Habash, Charbel El-Khaissi, Omer Goldman, Michael Gasser, William Lane, Matt Coler, Arturo Oncevay, Jaime Rafael Montoya Samame, Gema Celeste Silva Villegas, Adam Ek, Jean-Philippe Bernardy, Andrey Shcherbakov, Aziyana Bayyr-ool, Karina Sheifer, Sofya Ganieva, Matvey Plugaryov, Elena Klyachko, Ali Salehi, Andrew Krizhanovsky, Natalia Krizhanovsky, Clara Vania, Sardana Ivanova, Aelita Salchak, Christopher Straughn, Zoey Liu, Jonathan North Washington, Duygu Ataman, Witold Kieraś, Marcin Woliński, Totok Suhardijanto, Niklas Stoehr, Zahroh Nuriah, Shyam Ratan, Francis M. Tyers, Edoardo M. Ponti, Grant Aiton, Richard J. Hatcher, Emily Prud'hommeaux, Ritesh Kumar, Mans Hulden, Botond Barta, Dorina Lakatos, Gábor Szolnok, Judit Ács, Mohit Raj, David Yarowsky, Ryan Cotterell, Ben Ambridge, and Ekaterina Vylomova. 2021. SIGMORPHON 2021 shared task on morphological reinflection: Generalization across languages. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 229–259.

Annette Rios, Chantal Amrhein, Noëmi Aepli, and Rico Sennrich. 2021. On biasing transformer attention towards monotonicity. In *2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Farhan Samir and Miikka Silfverberg. 2022. One wug, two wug+ s transformer inflection models hallucinate affixes. In *Proceedings of the Fifth Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 31–40.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *54th Annual Meeting of the Association for Computational Linguistics*, pages 86–96.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017a. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017b. Attention is all you need. *Advances in neural information processing systems*, 30.

Ekaterina Vylomova, Jennifer White, Elizabeth Salesky, Sabrina J. Mielke, Shijie Wu, Edoardo Maria Ponti, Rowan Hall Maudslay, Ran Zmigrod, Josef Valvoda, Svetlana Toldova, Francis Tyers, Elena Klyachko, Ilya Yegorov, Natalia Krizhanovsky, Paula Czarnowska, Irene Nikkarinen, Andrew 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. SIGMORPHON 2020 shared task 0: Typologically diverse morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 1–39.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306.

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

Xiang Yu, Ngoc Thang Vu, and Jonas Kuhn. 2020. Ensemble self-training for low-resource languages: Grapheme-to-phoneme conversion and morphological inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 70–78.

A Lemma Overlap

Lemma overlap for small training data (in Table 3) and large training data (in Table 2) with the development and test sets. Lemma overlap is computed by dividing the number of examples, where the lemma occurs in the training set, with the total number of examples.

B Supplementary results

Table 4 shows the micro averaged inflection accuracy of each model on the development data.

	ang	ara	asm	evn	got	heb	hun	hye	kat	kaz	khk	kor	krl	lud	non	pol	poma	slk	tur	vep
dev	64.7	62.8	99.3	65.7	76.0	100.0	30.9	68.8	90.8	97.7	98.9	92.0	59.0	53.5	95.4	7.3	11.6	5.6	91.7	44.7
test	77.1	54.0	98.9	61.3	81.2	100.0	31.1	69.7	82.4	98.2	99.0	92.2	81.2	54.3	95.2	6.6	17.1	5.1	87.2	42.1

Table 2: Lemma overlap for the large training sets with the development and test data. Lemma overlap is computed as f/N, where f is the number of development/test examples, where the lemma is found in the training set and N is the total number of development/test examples.

	ang	ara	asm	bra	ckt	evn	gml	goh	got	guj	heb	hsb	hsi	hun	hye	itl	kat	kaz	ket
dev	14.2	13.3	45.4	25.8	27.3	36.7	100.0	78.7	13.7	83.7	45.5	20.0	73.3	3.1	14.7	27.8	52.3	97.7	57.6
test	19.0	8.9	45.9	30.7	34.8	29.8	100.0	80.6	16.0	81.8	43.6	16.2	63.3	4.0	15.2	25.5	28.4	98.2	44.5

	khk	kor	krl	lud	mag	nds	non	pol	poma	sjo	slk	slp	tur	vep
dev	26.1	23.1	10.1	12.5	36.7	90.7	38.9	0.5	1.5	32.3	0.6	65.0	50.5	7.2
test	24.7	23.7	16.1	9.7	35.3	92.1	40.4	0.9	1.6	25.3	0.4	72.2	45.4	5.0

Table 3: Lemma overlap for the small training sets with the development and test data.

Experiment	Small	Large
ST BASELINE	42.59	60.04
OUR BASELINE	43.52	67.37
HALL	49.28	67.49
COPY	52.41	68.57
COPY+SF	**53.36**	**68.99**
COPY+HALL	52.32	68.09

Table 4: Results on the development data under small and large data conditions. ST BASELINE refers to the official neural shared task baseline and "Our Baseline" to our baseline transformer with reverse positional encoding and type embeddings. SF refers to student forcing, HALL to data hallucination and COPY to lemma copying.