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).

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 reinfection, 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 understand-
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 \( (x_1, \ldots, x_T) \) 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 \( (h_1, \ldots, h_T) \). At generation step \( t \), the decoder reads the previously generated sequence \( (y_1, \ldots, y_{t-1}) \) to produce states \( (s_1, \ldots, s_{t-1}) \). 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:

\[
P_{\text{vocab}} = \text{softmax}(Vs_{t-1} + b) \tag{1}
\]

During training, the entire target sequence \( (y_1, \ldots, y_{T_y}) \) 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 \( p_{\text{gen}} \) – the probability of generating as opposed to copying – which acts as a soft switch between the two actions. \( p_{\text{gen}} \) 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.

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

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

\[
c_t = \sum_{i=1}^{T} a_t^i h_i \tag{3}
\]

with attention weights \( (a_1^t, \ldots, a_T^t) \). 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 \( p_{\text{gen}}, P_{\text{vocab}} \) produced by the transformer, and the attention weights of the last decoder layer \( (a_1^t, \ldots, a_T^t) \) to compute a distribution over the extended vocabulary:

\[
P(c) = p_{\text{gen}}P_{\text{vocab}}(c) + (1 - p_{\text{gen}})P_{\text{copy}}(c), \tag{4}
\]

with

\[
P_{\text{copy}}(c) = \sum_{x_i = c} a_t^i \tag{5}
\]

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,

¹https://github.com/AssafSinger94/sigmorphon-2020-infection
then $\sum_{i:x_i=c} a^i$ 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:

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

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.

### 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.

![Figure 2: English multitask training example (Task 0).](image)

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

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
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 (Chrupała, 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 Table 3. All three systems produce relatively similar results. NYU-CU Boulder-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-CU Boulder-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., 2017).

<table>
<thead>
<tr>
<th>System</th>
<th>Baseline 1</th>
<th>Baseline 2</th>
<th>Sub-1</th>
<th>Sub-2</th>
<th>Sub-3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>slots</td>
<td>macro</td>
<td>slots</td>
<td>macro</td>
<td>slots</td>
</tr>
<tr>
<td>Basque</td>
<td>30</td>
<td>0.0006</td>
<td>27</td>
<td>0.0006</td>
<td>30</td>
</tr>
<tr>
<td>Bulgarian</td>
<td>35</td>
<td>0.283</td>
<td>34</td>
<td>0.3169</td>
<td>35</td>
</tr>
<tr>
<td>English</td>
<td>4</td>
<td>0.656</td>
<td>4</td>
<td>0.662</td>
<td>4</td>
</tr>
<tr>
<td>Finnish</td>
<td>21</td>
<td>0.0533</td>
<td>21</td>
<td>0.055</td>
<td>21</td>
</tr>
<tr>
<td>German</td>
<td>9</td>
<td>0.2835</td>
<td>9</td>
<td>0.29</td>
<td>9</td>
</tr>
<tr>
<td>Kannada</td>
<td>172</td>
<td>0.1549</td>
<td>172</td>
<td>0.1512</td>
<td>172</td>
</tr>
<tr>
<td>Navajo</td>
<td>3</td>
<td>0.0323</td>
<td>3</td>
<td>0.0327</td>
<td>3</td>
</tr>
<tr>
<td>Spanish</td>
<td>29</td>
<td>0.2296</td>
<td>29</td>
<td>0.2367</td>
<td>29</td>
</tr>
<tr>
<td>Turkish</td>
<td>104</td>
<td>0.1421</td>
<td>104</td>
<td>0.1553</td>
<td>104</td>
</tr>
<tr>
<td>All</td>
<td>0.2039</td>
<td>0.2112</td>
<td>0.1749</td>
<td>0.1802</td>
<td>0.1765</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Multitask Train</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Hallucination</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>Development Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>88.20</td>
</tr>
<tr>
<td>Other</td>
<td>92.66</td>
</tr>
<tr>
<td>All</td>
<td>92.04</td>
</tr>
</tbody>
</table>

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. 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 pre-trained.

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
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


