Character-level Supervision for Low-resource POS Tagging

Katharina Kann[1], **Johannes Bjerva**[2],
Isabelle Augenstein[2], **Barbara Plank**[3], **Anders Søgaard**[2]
[1]Center for Data Science, New York University, USA
[2]Department of Computer Science, University of Copenhagen, Denmark
[3]Department of Computer Science, IT University of Copenhagen, Denmark
kann@nyu.edu

Abstract

Neural part-of-speech (POS) taggers are known to not perform well with little training data. As a step towards overcoming this problem, we present an architecture for learning more robust neural POS taggers by jointly training a hierarchical, recurrent model and a recurrent character-based sequence-to-sequence network supervised using an auxiliary objective. This way, we introduce stronger character-level supervision into the model, which enables better generalization to unseen words and provides regularization, making our encoding less prone to overfitting. We experiment with three auxiliary tasks: lemmatization, character-based word autoencoding, and character-based random string autoencoding. Experiments with minimal amounts of labeled data on 34 languages show that our new architecture outperforms a single-task baseline and, surprisingly, that, on average, raw text autoencoding can be as beneficial for low-resource POS tagging as using lemma information. Our neural POS tagger closes the gap to a state-of-the-art POS tagger (MarMoT) for low-resource scenarios by 43%, even outperforming it on languages with templatic morphology, e.g., Arabic, Hebrew, and Turkish, by some margin.

1 Introduction

POS tagging, i.e., assigning syntactic categories to tokens in context, is an important first step when developing language technology for low-resource languages. POS tags can provide an efficient inductive bias for modeling downstream tasks, especially if training data for these tasks are limited.

However, POS tagging can be very challenging if only a few labeled sentences are available. Previous work on POS tagging with limited or no annotated data comes in three flavors, e.g., (Yarowsky et al., 2001; Goldwater and Griffiths, 2007; Li et al., 2012; Biemann, 2012; Täckström et al., 2013; Dong et al., 2015; Agić et al., 2015): unsupervised POS induction, cross-lingual transfer, or, if some suitable data are available, supervised induction from small labeled corpora or dictionaries. This work focuses on the latter: We explore the effect of multi-task learning for building robust POS taggers for low-resource languages from small amounts of annotated data.

In low-resource settings, neural POS taggers have been observed to perform poorly compared to log-linear models. This is unfortunate, since neural POS taggers have other advantages, including being easily integrable into multi-task learning architectures, sidestepping feature engineering, and providing compact word-level and sentence-level representations. In this paper, we therefore take steps to bridge the gap to state-of-the-art taggers in such scenarios.

Specifically, we consider training neural POS taggers from 478 annotated tokens (the size of the smallest treebank in UD 2.0[1]). In such a setting, it is often useful to leverage data from other, related tasks (Bingel and Søgaard, 2017), if available. However, since for many low-resource languages such data is hard to find, we consider multi-task learning scenarios with no other sequence labeling data at hand:

(i) a scenario in which type-based morphological information is available, e.g., word-lemma pairs as can be found in standard dictionaries or UniMorph,[2]

[1]http://universaldependencies.org/
[2]http://unimorph.org/

Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP, pages 1–11
Melbourne, Australia July 19, 2018.

(ii) a scenario where we only rely on raw text corpora in the language, and

(iii) a scenario where we do not assume any additional data, but construct a synthetic auxiliary task instead.

In order to include secondary information such as word-lemma pairs into our model, we integrate a character-based recurrent sequence-to-sequence model into a hierarchical long short-term memory (LSTM) sequence tagger (cf. Figure 1). By formulating suitable auxiliary tasks (lemmatization, word autoencoding or random string autoencoding, respectively), we can include additional character-level supervision into our model via multi-task training.

Contributions. We present a novel architecture for inducing more robust neural POS taggers from small samples of annotated data in low-resource languages, combining a hierarchical, deep bi-LSTM sequence tagger with a character-based sequence-to-sequence model. Furthermore, we experiment with different choices of external resources and corresponding auxiliary tasks and show that autoencoding can be as efficient as an auxiliary task for low-resource POS tagging as lemmatization. Finally, we evaluate our models on 34 typologically diverse languages.

2 POS Tagging with Subword-level Supervision

Hierarchical POS tagging LSTMs that receive both word-level and subword-level input, such as Plank et al. (2016), are known to perform well on unseen words. This is due to their ability to associate subword-level patterns with POS tags. However, hierarchical LSTMs are also very expressive, and thus prone to overfitting. We believe that using subword-level auxiliary tasks to regularize the character-level encoding in hierarchical LSTMs is a flexible and efficient way to get the best of both worlds: such a model is still able to make predictions about unknown words, but the subword-level auxiliary task should prevent it from overfitting.

2.1 Hierarchical LSTMs with Character-level Decoding

Our proposed multi-task architecture is shown in Figure 1. For the hierarchical sequence labeling LSTM, we follow Plank et al. (2016): Our

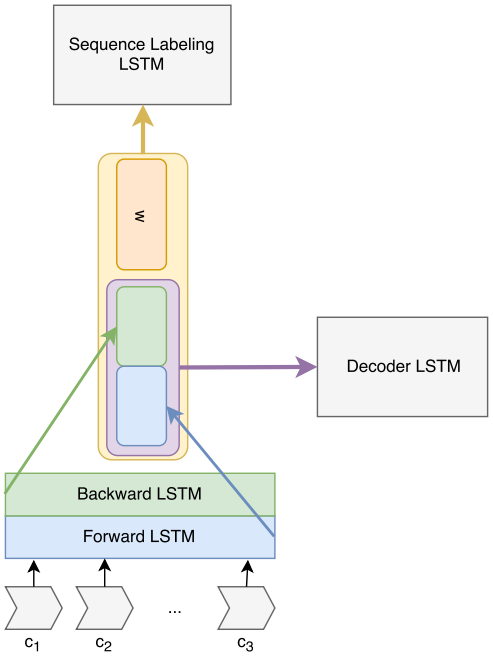


Figure 1: Our multi-task architecture, consisting of a shared character LSTM (down), as well as a sequence labeling (up) and a sequence-to-sequence (right) part.

subword-level LSTM is bi-directional and operates on the character level (Ling et al., 2015; Ballesteros et al., 2015). Its input is the character sequence of each input word, represented by the embedding sequence $c_1, c_2, \ldots, c_m$. The final character-based representation of each word is the concatenation of the two last LSTM hidden states:

$$v_{c,i} = conc(\text{LSTM}_{c,f}(c_{1:m}), \text{LSTM}_{c,b}(c_{m:1})) \quad (1)$$

Second, a context bi-LSTM operates on the word level. Like Plank et al. (2016), we use the term "context bi-LSTM" to denote a bidirectional LSTM which, in order to generate a representation for input element i, encodes all elements up to position i with a forward LSTM and all elements from n to i using a backward LSTM. For each sentence represented by embeddings $w_1, w_2, \ldots, w_n$, its input are the concatenation of the word embeddings with the outputs of the subword-level LSTM: $conc(w_1, v_{c,1}), conc(w_2, v_{c,2}) \ldots, conc(w_n, v_{c,n})$. The final representation which gets forwarded to the next part of the network is again the

concatenation of the last two hidden LSTM states:

$$v_{w,i} = conc(\text{LSTM}_{w,f}(conc(w, v_c)_{1:i}), \quad (2)$$
$$\text{LSTM}_{w,b}(conc(w, v_c)_{n:i}))$$

This is then passed on to a classification layer.

2.1.1 Character-based Decoding

We extend the network with a new component, a character-based sequence-to-sequence model. It consists of a bidirectional LSTM encoder which is connected to an LSTM decoder (Cho et al., 2014; Sutskever et al., 2014).

Encoding. The encoder corresponds to the character-level bi-directional LSTM described above and thus yields the representation

$$v_{c,i} = conc(\text{LSTM}_{c,f}(c_{1:m}), \quad (3)$$
$$\text{LSTM}_{c,b}(c_{m:1}))$$

for an input word embedded as $c_1, c_2, \ldots, c_m$. Parameters of the character-level LSTM are shared between the sequence labeling and the sequence-to-sequence part of our model.

Decoding. The decoder receives the concatenation of the last hidden states $v_{c,i}$ as input. In particular, we do not use an attention mechanism (Bahdanau et al., 2015), since our goal is not to improve performance on the auxiliary task, but instead to encourage the encoder to learn better word representations. The decoder is trained to predict each output character y_t dependent on $v_{c,i}$ and previous predictions y_1, ..., y_{t-1} as

$$p(y_t|\{y_1, ..., y_{t-1}\}, v_{c,i}) = g(y_{t-1}, s_t, v_{c,i}) \quad (4)$$

for a non-linear function g and the LSTM hidden state s_t. The final softmax output layer is calculated over the vocabulary of the language.

Joint model. Figure 1 shows how parameters are shared between the sequence labeling and the sequence-to-sequence components of our network. All model parameters, including all embeddings, are updated during training. Our model architecture is "symmetric", i.e., it does not distinguish between main and auxiliary tasks. However, we use early stopping on the development set of the main task, such that convergence is not guaranteed for the auxiliary tasks.

2.2 Multi-task Learning

We want to train our neural model jointly on (i) a low-resource main task, i.e., POS tagging, and (ii) an exchangeable auxiliary task (cf. §3). Therefore, we want to maximize the following joint log-likelihood:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(l,s)\in\mathcal{D}_{POS}} \log p_\theta\left(l \mid s\right) \quad (5)$$
$$+ \sum_{(in,out)\in\mathcal{D}_{aux}} \log p_\theta\left(out \mid in\right)$$

Here, $\mathcal{D}_{POS}$ denotes the POS tagging training data, with s being the input sentence and l the corresponding label sequence. $\mathcal{D}_{aux}$ is a placeholder for our auxiliary task training data with examples consisting of input in and output out. We experiment with three different auxiliary tasks, which will be described in the next section.

The set of model parameters $\boldsymbol{\theta}$ is the union of the set of parameters of the sequence labeling and the sequence-to-sequence part. Parameters of the character LSTM are shared between the main and the auxiliary task.

3 (Un)supervised Auxiliary Tasks

In this section, we will describe our three auxiliary tasks in more detail.

3.1 Random String Autoencoding

Random string autoencoding is a synthetic auxiliary task created for a setting in which we have no additional resources available. It consists of, given a random character sequence as input, reconstructing the same sequence in the output. Concretely, given the alphabet $\mathcal{A}_L$ of a language L, the task is to learn a mapping $r \mapsto r$ for $r \in \mathcal{A}_L^+$. Note that the random string r is in most cases not a valid word in L. Additionally, we prepend a special symbol $\mathcal{S}_r$ to the input which indicates the current task to the encoder, e.g., "OUT=AE" or "OUT=POS".

3.2 Word Autoencoding

Word autoencoding is a special case of the previous auxiliary task, in that we now use actual words in the language, e.g., from unlabeled corpora or dictionaries. As for random string autoencoding, the task consists of reproducing a given input character sequence in the output. As before, we additionally feed a special symbol $\mathcal{S}_w$ into the

model which signals the current task to the encoder. Our final training examples are of the form $(\mathcal{S}_w; w) \mapsto w$, where $w \in \mathcal{V}_L$ for the vocabulary $\mathcal{V}_L$ of L. Word autoencoding has been used as an auxiliary task before, e.g., (Vosoughi et al., 2016).

3.3 Lemmatization

Lemmatization is a task from the area of inflectional morphology. In particular, it is a special case of morphological inflection. Its goal is to map a given inflected word form to its lemma, e.g.,

$$\text{sueño} \mapsto \text{soñar}. \tag{6}$$

Sequence-to-sequence models have shown strong performances on morphological inflection (Aharoni et al., 2016; Kann and Schütze, 2016; Makarov et al., 2017). Therefore, when morphological dictionaries are available, we can easily combine a neural model for lemmatization with a POS tagger, using our architecture. Our intuition for this auxiliary task is that it should be possible to include morphological information into our character-based word representations.

Formally, the task can be described as follows. Let $\mathcal{A}_L$ be a discrete alphabet for language L and let $\mathcal{T}_L$ be a set of morphological tags for L. The morphological paradigm π of a lemma w in L is a set of pairs

$$\pi(w) = \Big\{ \big(f_k[w], t_k\big) \Big\}_{k \in T(w)} \tag{7}$$

where $f_k[w_\ell] \in \mathcal{A}_L^+$ is an inflected form, $t_k \in \mathcal{T}_\mathcal{L}$ is its morphological tag and $T(w)$ is the respective set of paradigm slots. Lemmatization consists of predicting the lemma w for an inflected form $f_k[w_\ell]$ in $\pi(w)$.

4 Experimental Setup

In this section, we will describe our experiments, including data, baselines, and hyperparameters.

4.1 Data

POS. The data for our POS tagging main task comes from the Universal Dependencies (*UD*) 2.0 collection (Nivre et al., 2007). We use the provided train/dev/test splits.

Since we use the official datasets from the SIGMORPHON 2017 shared task on universal morphological reinflection (Cotterell et al., 2017) for the lemmatization auxiliary task, we limit ourselves to the languages featured there. We simulate a low-resource setting by reducing all training sets to 478 tokens. Among our languages, this is the size of the smallest training set in UD 2.0.

Lemmatization. For the lemmatization auxiliary task, we make use of the word-lemma pairs in the training sets released for the SIGMORPHON 2017 shared task (Cotterell et al., 2017), which are subsets of the UniMorph data. In particular, there are three settings with different training sets per language: low (100 examples), medium (1,000 examples) and high (10,000 examples).

Word autoencoding. For the word autoencoding task, we use the inflected forms from the SIGMORPHON 2017 shared task dataset for each respective setting. Due to identical forms for different slot in the morphological paradigm of some lemmas, we might have duplicate examples in those datasets.

Random string autoencoding. For the random string autoencoding auxiliary task, we generate random character sequences to be used as training instances for our model's encoder-decoder part. In order to have the same amount of unique characters as with the other two auxiliary tasks, we use the character sets from the SIGMORPHON shared task vocabulary for each respective setting. We then uniformly draw characters from these sets and form strings of random lengths between 3 and 20 characters.

4.2 Baselines

TreeTagger. Since low-resource settings like the one considered here are known to be challenging for neural models, we employ TreeTagger (Schmid, 1995), a non-neural Markov model tagger, as our first baseline.

MarMoT. Our second non-neural baseline is the state-of-the-art tagger MarMoT (Müller et al., 2013), which is based on conditional random fields (CRFs, Lafferty et al. (2001)).

Single-task hierarchical LSTM. We compare all our results to a single-task baseline model, which corresponds largely to the architecture used by Plank et al. (2016) for POS tagging. We modify their original code by adding character dropout with a coefficient of 0.25 to improve regularization and make the baseline more comparable to and competitive with our models.

Multi-task baseline. We further compare to a multi-task architecture which jointly learns to predict the POS tag and the log-frequency of a word as suggested by Plank et al. (2016). The intuition described by the original authors is that the auxiliary loss, being predictive of word frequency, can improve the representations of rare words. Note that this baseline can easily be combined with our architecture. We leave the exploration of such a combination for future work.

4.3 Hyperparameters

For all networks, we use 300-dimensional character embeddings, 64-dimensional word embeddings and 100-dimensional LSTM hidden states. Encoder and decoder LSTMs have 1 hidden layer each. For training, we use ADAM (Kingma and Ba, 2014), as well as word dropout and character dropout, each with a coefficient of 0.25 (Kiperwasser and Goldberg, 2016). Gaussian noise is added to the concatenation of the last states of the character LSTMs for POS tagging. All models are trained using early stopping, with a minimum number of 75 (single-task and low), 30 (medium) or 20 (high) epochs and a maximum number of 300 epochs, which is never reached. We stop training if we obtain no improvement for 10 consecutive epochs. The best model on the development set is used for testing.

5 Results

The test results for all languages and settings are presented in Table 1.

Our first observation is that using 100 words of auxiliary task data seems to be sufficient, as we do not see consistent gains from adding more auxiliary task instances. This might be related to the very limited amount of POS tagging data we assume available; a too low main-auxiliary task data ratio probably inhibits further gains.

Second, we find that lemmatization and word autoencoding on average over all languages bring similar gains, differences are only between 0.0013 (medium) and 0.0021 (high) absolute accuracy. Comparing word and random string autoencoding, two observations can be made: in the low setting, differences are small, while random string autoencoding is the only task which performs worse in the high compared to the low setting. So the gap between the two autoencoding tasks grows bigger for larger auxiliary task data. This might be explained by random string autoencoding being helpful in order to get clearer distinctions between characters; however, this might as well destroy the model's ability to pick up on beneficial similarities.

Our third observation is that lemmatization and word autoencoding consistently outperform the auxiliary task of predicting log-frequencies as suggested in Plank et al. (2016) with up to 0.0081 (POS+AE, low/high) higher absolute accuracy; random string autoencoding performs 0.0079 better in the low setting. We may thus conclude that, in our setting, auxiliary tasks with additional character-level supervision are more beneficial.

Fourth, both non-neural baselines outperform the single-task neural model. Adding auxiliary tasks leads to a higher performance (averaged over languages) than TreeTagger. MarMoT is the overall best performing model. However, for some individual languages, the neural model obtains higher accuracies, e.g., for Bulgarian, Dutch, or Romanian. In particular, our approach is stronger for languages with templatic morphology, e.g., Arabic, Hebrew, or Turkish. This emphasizes the importance of neural approaches for the task.

Finally, we look at differences between auxiliary tasks for individual languages. Here, we notice that autoencoders often outperform lemmatization for agglutinative languages. An explanation for this might be that agglutinative morphology is harder to learn, and the chance of overfitting on a small sample is therefore higher.

6 Analysis

6.1 Error Analysis

Table 2 lists the F_1-scores of our models across POS tags, compared to the single-task baseline.

Our first observation is that the decrease in performance from training on *more* random strings, is relatively equal across tags, with the exception of DET, PUNCT and X; tokens that consist of very few, fixed characters. We also note that all our models with character-level supervision get worse at predicting numerals. In contrast, ADP, AUX, CCONJ and PUNCT always benefit from a character-based auxiliary task. Generally, the POS taggers trained on small amounts of data are challenged by rare syntactic categories such as interjections and the miscellaneous category X.

	high			medium			low			baselines			
language	POS+ Lemma	POS+ AE	POS+ AE-Random	POS+ Lemma	POS+ AE	POS+ AE-Random	POS+ Lemma	POS+ AE	POS+ AE-Random	POS+ LogFrequ	POS	TreeTagger	MarMoT
arabic	**.6900(.01)**	.6862(.01)	.6747(.01)	.6638(.02)	.6737(.01)	.6635(.00)	.6735(.01)	.6680(.01)	.6731(.01)	.6583(.02)	.6545(.01)	.6047(.00)	.6398(.00)
basque	.5925(.01)	.6231(.01)	.5866(.01)	.6088(.01)	.6291(.01)	.6170(.01)	.6332(.01)	.6262(.01)	.6363(.00)	.6058(.01)	.6217(.02)	.6143(.00)	**.6862(.00)**
bulgarian	.6139(.01)	.6344(.01)	.5954(.01)	.6239(.01)	.6343(.00)	.6153(.02)	.6267(.01)	.6410(.01)	**.6431(.01)**	**.6431(.01)**	.6272(.01)	.6208(.00)	.6067(.00)
catalan	.7104(.01)	.7139(.01)	.7051(.00)	.7175(.01)	.7212(.01)	.7116(.01)	.7279(.01)	.7352(.01)	.7232(.01)	.7024(.01)	.7197(.01)	.7525(.00)	**.7824(.00)**
czech	.6227(.01)	.6063(.01)	.5574(.02)	.5923(.01)	.5953(.00)	.5742(.01)	.6131(.02)	.6048(.00)	.6003(.01)	.5784(.01)	.5897(.02)	.6417(.00)	**.6720(.00)**
danish	.6368(.00)	.6322(.01)	.6137(.01)	.6431(.01)	.6434(.00)	.6307(.01)	.6375(.01)	.6378(.00)	.6378(.00)	.6431(.01)	.6346(.03)	.6346(.00)	**.6703(.00)**
dutch	**.5901(.01)**	.5835(.01)	.5533(.01)	.5837(.00)	.5778(.00)	.5689(.01)	.5810(.01)	.5807(.00)	.5812(.01)	.5559(.01)	.5724(.01)	.5406(.00)	.5866(.00)
english	.5784(.01)	.5723(.01)	.5183(.01)	.5747(.01)	.5658(.01)	.5423(.00)	.5585(.03)	.5836(.01)	.5857(.00)	.5751(.02)	.5822(.02)	.6019(.00)	**.6507(.00)**
estonian	.5386(.01)	.5456(.01)	.5224(.01)	.5438(.01)	.5323(.01)	.5173(.01)	.5384(.01)	.5474(.01)	.5333(.02)	.5388(.01)	.5391(.02)	.5632(.00)	**.5851(.00)**
finnish	**.5748(.01)**	.5678(.01)	.5403(.02)	.5599(.01)	.5632(.01)	.5468(.01)	.5637(.01)	.5616(.01)	.5634(.00)	.5424(.01)	.5525(.01)	.5602(.00)	.5743(.00)
french	.6922(.01)	.6898(.01)	.6720(.00)	.6855(.01)	.6874(.00)	.6876(.01)	.7048(.00)	.6981(.00)	.6924(.01)	.6858(.01)	.6826(.01)	.5931(.00)	**.7084(.00)**
german	.6804(.01)	.6742(.00)	.5739(.02)	.6887(.00)	.6639(.01)	.6190(.01)	.6768(.01)	.6830(.01)	.6687(.01)	.6895(.01)	.6711(.01)	.5912(.00)	**.7370(.00)**
hebrew	.6764(.00)	**.6838(.00)**	.6827(.01)	.6734(.01)	.6781(.00)	.6672(.00)	.6761(.01)	.6825(.00)	.6796(.01)	.6776(.01)	.6655(.01)	.6147(.00)	.6705(.00)
hindi	.5954(.01)	**.6062(.01)**	.5874(.01)	.5914(.01)	.5984(.01)	.5839(.01)	.5989(.01)	.6046(.01)	.6001(.01)	.5992(.01)	.5791(.02)	.5784(.00)	.5943(.00)
hungarian	.5820(.01)	.5853(.01)	.5647(.01)	.5811(.01)	.5836(.01)	.5712(.01)	.5907(.01)	.5897(.01)	.5991(.00)	.5824(.01)	.5825(.02)	.6352(.00)	**.6651(.00)**
irish	**.6800(.01)**	.6715(.01)	.6573(.00)	.6741(.00)	.6649(.00)	.6593(.01)	.6735(.00)	.6743(.00)	.6790(.00)	.6724(.01)	.6699(.00)	.6511(.00)	.6729(.00)
italian	.7150(.00)	.7123(.01)	.6992(.00)	.7105(.00)	.7032(.00)	.7012(.00)	.7012(.01)	.7054(.01)	.7053(.00)	.7076(.01)	.6902(.02)	.6959(.00)	**.7280(.00)**
latin	.5950(.00)	.5853(.01)	.5633(.01)	.6026(.01)	.5953(.00)	.5863(.00)	.5998(.01)	.6068(.01)	.5992(.01)	.6069(.00)	.6046(.01)	.6234(.00)	**.6312(.00)**
latvian	.5446(.01)	.5557(.01)	.5288(.01)	.5406(.01)	.5453(.01)	.5303(.01)	.5344(.01)	.5356(.01)	.5408(.01)	.5382(.00)	.5300(.01)	**.5984(.00)**	.5773(.00)
lithuanian	.5347(.01)	.5292(.02)	.4936(.01)	.5123(.01)	.5192(.01)	.5213(.01)	.5230(.01)	.5313(.01)	.5327(.01)	.5238(.01)	.4721(.01)	.5783(.00)	**.5840(.00)**
n.-bokmaal	.5388(.00)	.5399(.01)	.5048(.01)	.5352(.01)	.5368(.01)	.5280(.01)	.5365(.01)	.5400(.01)	.5437(.01)	.5300(.01)	.5016(.02)	**.5737(.00)**	.5658(.00)
n.-nynorsk	.6172(.01)	.6223(.01)	.6043(.01)	.6200(.01)	.6228(.01)	.6167(.01)	.6256(.01)	**.6263(.01)**	.6232(.01)	.6238(.01)	.6130(.01)	.6142(.00)	.6168(.00)
persian	.7419(.01)	.7438(.01)	.7287(.01)	.7312(.01)	.7277(.01)	.7330(.00)	.7332(.00)	.7352(.00)	.7340(.00)	.7278(.01)	.7116(.01)	.7339(.00)	**.7539(.00)**
polish	.6407(.00)	.6423(.01)	.5960(.01)	.6314(.01)	.6243(.01)	.6162(.01)	.6448(.00)	.6460(.01)	.6400(.01)	.6243(.01)	.6489(.01)	**.6712(.00)**	.6700(.00)
portuguese	.6886(.00)	.6865(.00)	.6597(.01)	.6786(.01)	.6775(.01)	.6692(.01)	.6784(.00)	.6815(.00)	.6920(.01)	.6482(.03)	.6408(.02)	.6377(.00)	**.7135(.00)**
romanian	.6029(.01)	**.6156(.01)**	.5956(.01)	.6028(.01)	.6117(.01)	.5969(.00)	.6027(.01)	.6135(.01)	.6145(.01)	.6062(.01)	.6051(.00)	.5993(.00)	.5740(.00)
russian	.6807(.01)	.6817(.01)	.6283(.00)	.6860(.02)	.6607(.01)	.6504(.01)	.6752(.01)	.6661(.01)	.6661(.01)	.6644(.01)	.6241(.05)	.6105(.00)	**.7281(.00)**
slovak	.6169(.01)	.6327(.01)	.5835(.01)	.6274(.01)	.6203(.01)	.6337(.02)	.6585(.01)	.6378(.01)	.6443(.01)	.6536(.02)	.6264(.02)	.6642(.00)	**.6672(.00)**
slovene	.6364(.00)	.6414(.00)	.6093(.01)	.6343(.01)	.6223(.01)	.6070(.01)	.6358(.00)	.6375(.01)	.6350(.01)	.6105(.01)	.5922(.02)	**.6561(.00)**	.6046(.00)
spanish	.6962(.00)	.6891(.01)	.6624(.02)	.6844(.01)	.6797(.01)	.6659(.01)	.6818(.01)	.6939(.01)	.6900(.01)	.6917(.01)	.6724(.01)	.6933(.00)	**.7578(.00)**
swedish	.6127(.01)	.6261(.01)	.5909(.01)	.6254(.00)	.6262(.00)	.6194(.01)	.6193(.01)	.6254(.01)	.6254(.00)	.6274(.01)	.6117(.02)	.6290(.00)	**.6304(.00)**
turkish	.6067(.01)	.6007(.01)	.5725(.01)	.5846(.01)	.5879(.00)	.5792(.01)	.6042(.01)	.6003(.01)	.6040(.01)	.6087(.01)	.5891(.01)	**.6107(.00)**	.6025(.00)
ukrainian	.5910(.01)	.5946(.00)	.5755(.00)	.5826(.01)	.5750(.00)	.5668(.01)	.5878(.02)	.5895(.01)	.5894(.01)	.5869(.02)	.5932(.01)	.5297(.00)	**.6125(.00)**
urdu	.6589(.01)	.6697(.01)	.6335(.00)	.6572(.01)	.6599(.00)	.6465(.01)	.6578(.01)	.6526(.01)	.6600(.01)	.6395(.02)	.6096(.03)	.5229(.00)	**.6776(.00)**
average	.6286(-)	.6307(-)	.6010(-)	.6251(-)	.6238(-)	.6131(-)	.6287(-)	.6307(-)	.6305(-)	.6226(-)	.6141(-)	.6188(-)	**.6529**(-)

Table 1: Averaged accuracies and standard deviations over 5 training runs on UD 2.0 test sets, with 478 tokens of POS-annotated data and varying amounts of data for the auxiliary task (low, medium and high). Best result for each language in bold. Autoencoding and lemmatization are *on par* across the board, and with 100 training sentences (low), random autoencoding is also competitive.

6.2 Why does Random String Autoencoding Help?

In the low setting, i.e., when using only 100 auxiliary task examples, autoencoding, especially of random strings, works better than or equally well as lemmatization for highly agglutinative languages such as Basque, Finnish, Hungarian, and Turkish. Further, while random string autoencoding is in general less efficient than autoencoding or lemmatization, it performs on par with these auxiliary tasks in the set-up with least auxiliary task data. However, this raises the question why random string autoencoding does work at all for a POS tagging main task. We offer two potential explanations:

General properties of the auxiliary tasks. Bingel and Søgaard (2017) showed that multi-task learning is more likely to be helpful when the auxiliary loss does not plateau earlier than the main loss. Figure 2 presents the loss curves for one model for each of four randomly selected languages (the corresponding plots for the remaining languages look similar). They show exactly the patterns found to be predictive of multi-task learning gains by Bingel and Søgaard (2017), who offer the explanation that when the auxiliary loss does not plateau before the target task, it can help the model out of local minima during training.

Preventing character collisions. A random string autoencoder needs to memorize the input string. This means encoding which characters are at what position in the input sequence. Jointly learning a random string autoencoder thus forces a model to make it easy to differentiate between characters, pushing them apart in vector space. See Table 3 for the average character distances and Table 4 for the minimum character distances across languages for our three systems (low setting) and our single-task baseline. For each system, the score is obtained by first calculating the average distance between all characters or, respectively, finding the minimum distance between any two characters for each language, and then computing the average across all languages.

Tag	high			medium			low			POS
	Δ Lemma	**Δ AE**	**Δ AE-Rand.**	**Δ Lemma**	**Δ AE**	**Δ AE-Rand.**	**Δ Lemma**	**Δ AE**	**Δ AE-Rand.**	
ADJ	-0.0090	0.0083	-0.0555	-0.0167	-0.0017	-0.0290	0.0139	0.0145	0.0045	0.4756
ADP	0.0308	0.0288	0.0042	0.0440	0.0345	0.0205	0.0356	0.0313	0.0293	0.7687
ADV	-0.0008	-0.0133	-0.0426	-0.0180	-0.0108	-0.0233	0.0166	0.0209	0.0193	0.2958
AUX	0.0610	0.0537	0.0203	0.0479	0.0264	0.0102	0.0250	0.0423	0.0300	0.6172
CCONJ	0.0849	0.0511	0.0400	0.0579	0.0449	0.0512	0.0646	0.0560	0.0606	0.7594
CONJ	0.0230	-0.0588	-0.0316	-0.0342	0.0674	0.1477	-0.0128	-0.1287	-0.0294	0.6617
DET	0.0227	0.0234	0.0048	0.0178	0.0210	-0.0151	0.0063	0.0046	-0.0037	0.6938
INTJ	-0.0204	-0.0072	-0.0022	-0.0229	-0.0133	-0.0086	-0.0217	-0.0115	-0.0166	0.0806
NOUN	0.0015	0.0127	-0.0226	-0.0009	0.0072	-0.0055	0.0099	0.0147	0.0076	0.5457
NUM	-0.0537	-0.0777	-0.1332	-0.1389	-0.1626	-0.1372	-0.0830	-0.0633	-0.0885	0.5965
PART	0.0313	0.0195	-0.0252	0.0089	-0.0280	-0.0558	-0.0221	-0.0046	-0.0013	0.6719
PRON	0.0532	0.0389	-0.0084	0.0435	0.0297	-0.0059	0.0346	0.0391	0.0368	0.5189
PROPN	-0.0374	-0.0529	-0.1133	-0.0682	-0.0503	-0.0684	-0.0318	-0.0197	-0.0249	0.4271
PUNCT	0.0342	0.0204	0.0192	0.0188	0.0187	0.0186	0.0199	0.0185	0.0142	0.9299
SCONJ	0.0243	0.0346	0.0000	0.0288	0.0116	-0.0073	0.0205	0.0102	0.0166	0.5708
SYM	-0.0495	-0.1380	-0.0482	-0.0590	0.0394	0.0072	0.1261	0.1738	0.1496	0.6437
VERB	0.0334	0.0397	-0.0268	0.0314	0.0234	0.0000	0.0416	0.0224	0.0378	0.4370
X	0.0829	0.0754	0.0262	0.0194	0.0175	0.0092	0.0126	0.0023	0.0019	0.1322

Table 2: F-score deltas between the neural single-task baseline (POS) and our multi-task systems.

System	Average Distance		
	low	medium	high
POS+Lemmatization	0.928	0.976	0.964
POS+AE	0.923	0.965	0.960
POS+AE-Random	0.913	0.956	0.996
POS		0.881	

Table 3: Average character embedding distances, averaged over all languages.

System	Minimum Distance		
	low	medium	high
POS+Lemmatization	0.031	0.027	0.074
POS+AE	0.032	0.031	0.092
POS+AE-Random	0.033	0.032	0.104
POS		0.018	

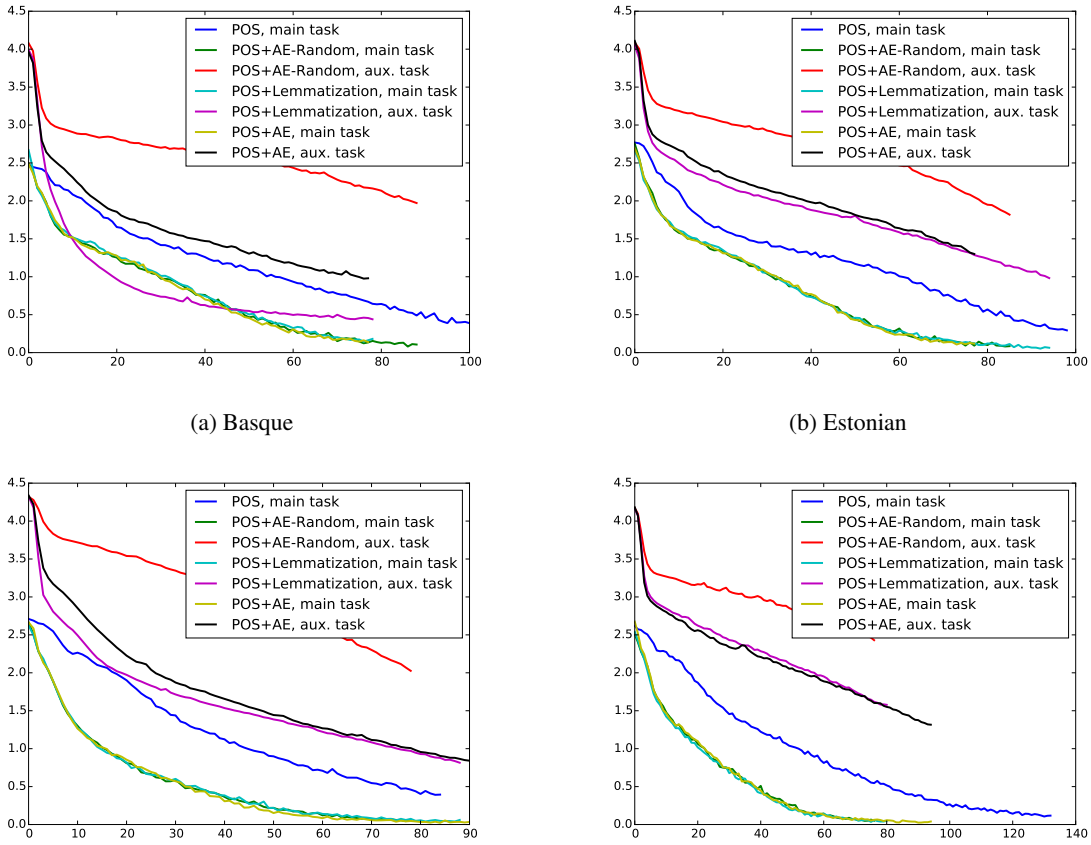
Table 4: Minimum character embedding distances, averaged over all languages.

In small sample regimes, pushing individual characters further apart is a potential advantage, since character collisions can be hurtful at inference time. We note how this is analogous to feature swamping of covariate features, as described in Sutton et al. (2006). Sutton et al. (2006) use a group lasso regularizer to prevent feature swamping. In the same way, we could also detect distributionally similar characters and use a group lasso regularizer to prevent covariate characters to swamp each other. However, this effect can potentially also hurt performance if done in an uninformed way. We intuit that this makes it also impossible for the model to learn useful similarities between characters (random string autoencoding in the high setting has a minimum distance of 0.104 compared to 0.018 for the single-task model). This might explain the performance gap between random string encoding and the other two auxiliary tasks for the high setting.

7 Related Work

POS tagging. POS tagging and other NLP sequence labeling tasks have been successfully approached using bidirectional LSTMs (Wang et al., 2015; Plank et al., 2016; Yang et al., 2016). Although previous work using such architectures often relies on massive datasets, Plank et al. (2016) show that bi-LSTMs in particular are not as reliant on large amounts of data in a sequence labeling scenario as previously assumed. Furthermore, their model is also a multi-task model, being trained jointly on predicting the POS and the log-frequency of a word. Their architecture obtained state-of-the-art results for POS tagging in several languages. Hence, in the low-resource setting considered here, we build upon the architecture developed by Plank et al. (2016), and extend it to a multi-task architecture involving sequence-to-sequence learning. Note though that in contrast to our setup, their tasks are both sequence-labeling tasks and using the same input for both tasks.

The same holds true for the multi-task model by Rei (2017), which is used to investigate how an additional language modeling objective could improve performance for sequence labeling without any need for additional training data. He reported



(a) Basque

(b) Estonian

(c) Hindi

(d) Swedish

Figure 2: Learning curve plots for four randomly selected languages, low setting.

gains for all investigated tasks, including POS. Finally, Gillick et al. (2016) present a multi-lingual model based on ideas from multi-task training, with each language constituting a separate task.

Multi-task learning in NLP. Neural networks make multi-task learning via (hard) parameter sharing particularly easy; thus, different task combinations have been investigated exhaustively. For sequence labeling, many combinations of tasks have been explored, e.g. by Søgaard and Goldberg (2016); Martínez Alonso and Plank (2017); Bjerva et al. (2016); Bjerva (2017a,b); Augenstein and Søgaard (2018). An analysis of task combinations is performed by Bingel and Søgaard (2017). Ruder et al. (2017) present a more flexible architecture, which learns what to share between the main and auxiliary tasks. Augenstein et al. (2017) combine multi-task learning with semi-supervised learning for strongly related tasks with different output spaces.

However, work on combining sequence labeling main tasks and sequence-to-sequence auxiliary tasks is harder to find. Dai and Le (2015) pretrain an LSTM as part of a sequence autoencoder on unlabeled data to obtain better performance on a sequence classification task. However, they report poor results for joint training. We obtain different results: even simple sequence-to-sequence tasks can indeed be beneficial for the sequence labeling task of low-resource POS tagging. This might be due to differences in the architectures or tasks.

Cross-lingual learning. Even though we do not employ cross-lingual learning in this work, we consider it highly relevant for low-resource settings and, thus, want to mention some important work here. Cross-lingual approaches have been used for a large variety of tasks, e.g., automatic speech recognition (Huang et al., 2013), entity recognition (Wang and Manning, 2014), language modeling (Tsvetkov et al., 2016), or parsing (Cohen et al., 2011; Søgaard, 2011; Naseem et al., 2012; Ammar et al., 2016). In the realm of sequence-to-sequence models, most work has been done for machine translation (Dong et al.,

2015; Zoph and Knight, 2016; Ha et al., 2016; Johnson et al., 2017). Another example is a character-based approach by Kann et al. (2017) for morphological generation.

8 Conclusion

We explored multi-task setups for training robust POS taggers for low-resource languages from small amounts of annotated data. In order to add additional character-level supervision into a hierarchical recurrent neural model, we introduced a novel network architecture. We considered different available types of external resources (word-lemma pairs, unlabeled corpora, or none) and employed corresponding auxiliary tasks (lemmatization, word autoencoding, or the artificial task of random string autoencoding) as well as varying amounts of auxiliary task data. While we did not find a systematic superior performance of models which were trained with lemmatization as an auxiliary task, the results confirmed our hypothesis that additional subword-level supervision improves POS taggers for resource-poor languages.

Acknowledgments

We would like to thank Paulina Grnarova and Rodrigo Nogueira for their helpful feedback. Isabelle Augenstein is partly supported by Eurostars grant Number E10138.

References

Željko Agić, Dirk Hovy, and Anders Søgaard. 2015. If all you have is a bit of the bible: Learning pos taggers for truly low-resource languages. In *ACL-IJCNLP*.

Roee Aharoni, Yoav Goldberg, and Yonatan Belinkov. 2016. Improving sequence to sequence learning for morphological inflection generation: The BIU-MIT systems for the SIGMORPHON 2016 shared task for morphological reinflection. In *SIGMORPHON*.

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah Smith. 2016. Many languages, one parser. *TACL*, 4:431–444.

Isabelle Augenstein, Sebastian Ruder, and Anders Søgaard. 2017. Multi-task learning of keyphrase boundary detection. In *ACL*.

Isabelle Augenstein and Anders Søgaard. 2018. Multi-task learning of pairwise sequence classification tasks over disparate label spaces. In *NAACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *EMNLP*.

Chris Biemann. 2012. Unsupervised part-of-speech tagging. In *Structure Discovery in Natural Language*, pages 113–144. Springer.

Joachim Bingel and Anders Søgaard. 2017. Identifying beneficial task relations for multi-task learning in deep neural networks. In *EACL*.

Johannes Bjerva. 2017a. *One Model to Rule them all: Multitask and Multilingual Modelling for Lexical Analysis*. Ph.D. thesis, University of Groningen.

Johannes Bjerva. 2017b. Will my auxiliary tagging task help? Estimating auxiliary tasks effectivity in multi-task learning. In *NoDaLiDa*.

Johannes Bjerva, Barbara Plank, and Johan Bos. 2016. Semantic tagging with deep residual networks. In *COLING*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *SSST*.

Shay B Cohen, Dipanjan Das, and Noah A Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *EMNLP*.

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. The CoNLL-SIGMORPHON 2017 shared task: Universal morphological reinflection in 52 languages. In *CoNLL-SIGMORPHON*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *ACL-IJCNLP*.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2016. Multilingual language processing from bytes. In *NAACL-HLT*.

Sharon Goldwater and Tom Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *ACL*.

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. 2016. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv:1611.04798*.

Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and n Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *IEEE*.

Melvin Johnson, Mike Schuster, Quoc Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernand a Vigas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *TACL*, 5:339–351.

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. 2017. One-shot neural cross-lingual transfer for paradigm completion. In *ACL*.

Katharina Kann and Hinrich Schütze. 2016. Single-model encoder-decoder with explicit morphological representation for reinflection. In *ACL*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Shen Li, Joao V Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *EMNLP*.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Peter Makarov, Tatiana Ruzsics, and Simon Clematide. 2017. Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection. In *CoNLL-SIGMORPHON*.

Héctor Martínez Alonso and Barbara Plank. 2017. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In *EACL*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *EMNLP*.

Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *ACL*.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *EMNLP-CoNLL*.

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

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *ACL*.

Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. 2017. Sluice networks: Learning what to share between loosely related tasks. *arXiv:1705.08142*.

Helmut Schmid. 1995. Improvements in part-of-speech tagging with an application to German. In *SIGDAT*.

Anders Søgaard. 2011. Data point selection for cross-language adaptation of dependency parsers. In *ACL-HLT*.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2006. Reducing weight undertraining in structured discriminative learning. In *NAACL*.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *TACL*, 1:1–12.

Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *NAACL-HLT*.

Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. 2016. Tweet2Vec: Learning tweet embeddings using character-level CNN-LSTM encoder-decoder. In *SIGIR*.

Mengqiu Wang and Christopher D Manning. 2014. Cross-lingual pseudo-projected expectation regularization for weakly supervised learning. *TACL*, 2:55–66.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. A unified tagging solution: Bidirectional LSTM recurrent neural network with word embedding. *arXiv:1511.00215*.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv:1603.06270*.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *HLT*.

Barret Zoph and Kevin Knight. 2016. Multi-source neural translation. In *NAACL-HLT*.