Part of Speech Tagging with Multi-Task Learning and Tokens Morphological Features

Ido Calman
308353499
calman.ido@gmail.com

Uri Avron
308046994
uriavron@gmail.com

Ofri Kleinfeld
302893680
ofrik@mail.tau.ac.il

Abstract

The Multi-task learning (MTL) approach has led to successes in many Natural Language Processing (NLP) tasks recently. This paper attempts to find out whether this approach improves the state-of-the-art results for Part of Speech (POS) tagging, Which is one of the most fundamental tasks in the NLP field. Even though current approaches achieve high accuracy on test sets, we believe that there is room to improve performance based on more strict evaluation methods, such as accuracy on tokens that did not appear in the training data or had more than one possible tag, and evaluation over uncorrelated test data.

1 Introduction

POS Tagging is a basic methodology of understanding the functional role of a token in a certain sentence. Many POS taggers report relatively high accuracy. The Stanford POS Tagger (Klein, 2003), TnT POS Tagger (Brants, 2000) and SVMTool POS Tagger (Gimenez, 2004) have reported their average token-level accuracies at over 96%.

However, when evaluating over average of the whole test set, these results may be misleading. In most datasets that are currently in extensive use such as the Universal Dependencies (http://universaldependencies.org/) around 90% of the tokens that appear in the training set would appear in the test set, and only around 15% of the tokens were observed with more than one POS label.

We shall attempt to adapt the novel approaches in NLP to improve the state-of-the-art Bi-directional Long Short Term Memory (BiLSTM) results (Plank et al, 2016), based on a MTL architecture (Søgaard, 2016) that achieved proven results in syntactic chunking and CCG supertagging. In our case, the lower level layers should hold a token-level information of the morphological features, such as gender, definite, tense and polarity.

Intuitively, the token's morphological information should hold some hints regarding the token's functional role in the sentence. Moreover, those features are being predicted at high accuracy, which hold to the principles of deep MTL architecture that achieved good results on other tasks.

Finally, we shall examine how a certain learned feature at a lower level affects the prediction of unseen and ambiguous tokens, and compare it to the BiLSTM baseline, alongside with an MTL architecture that learns all the available features at lower level layers.

2 Sequence Tagging with MTL BiLSTMs

Notation We denote Σ the set of all valid tokens in a language L, and T the set of all possible POS tags in a language. A sequence tagging is a process that receives a sentence $s \in \Sigma^m$ and produces for each sentence token s_i, a tag $t_i \in T$, where m is the sequence length which is predefined for the sake of enabling parallel computation (See 3.2). We use $F_\Theta(\cdot)$ as a function parametrized with parameters Θ. We use a special token $\omega \in \Sigma$ for padding, with a correspondent tag $\Omega \in T$. (See 2.1). For each morphological feature f we use T_f the set of all available tags of the feature. We also write the operator $\circ$ for vector concatenation.

2.1 Deep BiLSTMs

Each token s_i in the sentence s is a one-hot encoding of a unique index in a vector of size $|\Sigma|$ and we use a fully connected layer to transform the arbitrary index vectors into some meaningful representation (Tomas Mikolov et al, 2013) in the embedding space $\mathbb{E} \subseteq \mathbb{R}^e$ using the matrix $E \in \mathbb{R}^{e \times |\Sigma|}$. so:

$$e_i = E \cdot s_i \in \mathbb{R}^e$$

We use a specific Recurrent Neural Network implementation called long-short-term-memory (Hochreiter and Schmidhuber, 1997) in a bi-directional fashion (Schuster and Kuldip, 1997) to get a representation of the input sentence s, which is a series of m vectors, one for each token in the sentence, $h_1, \ldots, h_m \in \mathbb{R}^{2d}$ where d is the pre-defined hidden layer size.

Our BiLSTM layer is fully connected to a layer of size $|T|$ to produce a probability distribution over predicted POS tags using the matrix $P \in \mathbb{R}^{|T| \times 2d}$ For brevity, we will treat the LSTM as a black-box. By bi-direction we mean that our representations $h_1, \ldots, h_m$ are composed of two $LSTM$ architectures $LSTM_f, LSTM_b$:

$$\begin{aligned} h_i &= BiLSTM(e_i) \\ &= LSTM_f(e_i) \circ LSTM_b(e_i) \end{aligned}$$

This way the vector h_i is a representation of both the start-to-end and the end-to-start context of the token s_i. To increase regularization we use a dropout layer (Srivastava et al, 2014) with rate α. So our classification process works in the following manner to result in a probability distribution vector over the tag space:

$$\begin{aligned} \hat{y}_i &= softmax((P_\Theta \cdot h_i)) \\ &= softmax(P_\Theta \cdot BiLSTM_\Theta(e_i)) \\ &= softmax(P_\Theta \cdot BiLSTM_\Theta(E_\Theta \cdot s_i)) \end{aligned}$$

All the parameters Θ (the embedding vectors for the different vocabulary items, the parameters of the BiLSTM and the parameters of the classification function) are trained jointly in order to minimize the tagging loss over a sentence. We use the sum of cross-entropy over token loss:

$$\begin{aligned} J(s) &= \sum_{i=1}^{m} CE(y_i, \hat{y}_i) \\ &= -\sum_{i=1}^{m} \sum_{t=1}^{|T|} (y_i)_t \cdot \log{(\hat{y}_i)_t} \end{aligned}$$

Where $y_i \in \mathbb{R}^{|T|}$ is a one-hot encoding vector with entry 1 in the index corresponding to s_i real tag. This process provides us with a distribution over all possible tags, so finally we infer the tagging by applying **greedy** sequence tagging approach, that is:

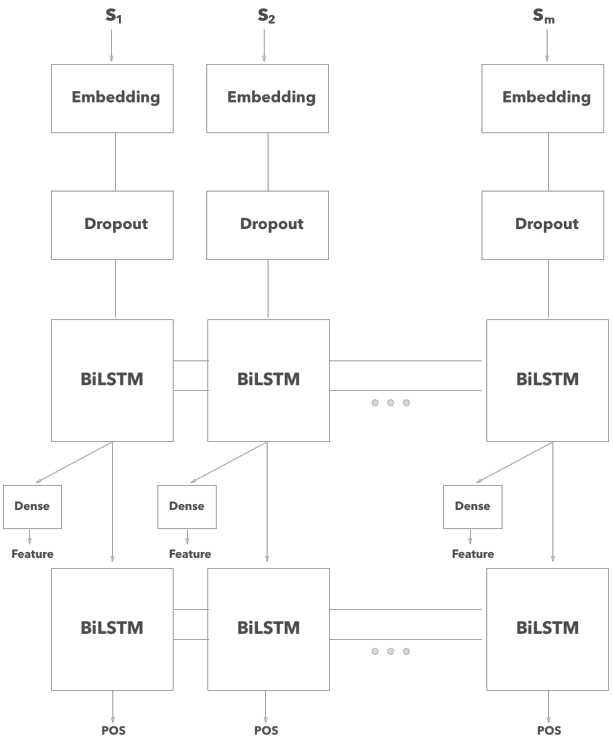

Figure 1: MTL with one low-level task

$$tag(s_i) = \text{argmax}\, \hat{y}_i$$

2.2 Single Feature MTL BiLSTMs

MTL architectures provide a way to store information about a single input with multiple outputs (Ruder, 2017). In our case, within a single input training process we derive gradients twice; once for the feature tag (i.e. lower-level task) and once for the POS tag (i.e. higher-level task).

Intuitively, even though the low-level task (such as tense detection) is different than the high-level task (i.e. POS tagging), the two share some sub-structures. For example, if our model successfully detects for a token that it's tense is 'past', it should be less likely to tag the word as 'adjective', which is a tag that is invariant to tense.

We embrace the partial deriving method (Søgaard, 2016), that is, deriving gradients for the whole architecture had we seen a POS tag and only deriving the shared layer gradients had we seen a feature tag.

For the low-level task prediction, the hidden BiLSTM output for the token s_i, h_i is fully connected via a matrix $F \in \mathbb{R}^{|T_f| \times 2d}$ to an output-space of dimension T_f. So here:

$$\hat{f}_i = softmax((F_\Theta \cdot h_i)) \\ = softmax(F_\Theta \cdot BiLSTM_\Theta(E_\Theta \cdot s_i))$$

But, for the high-level task, we stack another $BiLSTM$ layer which we will denote $BiLSTM'$. Then:

$$\hat{y}_i = softmax((P_\Theta \cdot h'_i)) \\ = softmax(P_\Theta \cdot BiLSTM'_\Theta(h_i)) \\ = softmax(P_\Theta \cdot BiLSTM'_\Theta \\ (BiLSTM_\Theta(E_\Theta \cdot s_i)))$$

Note that when training over a high-level task we derive the whole model, whilst training over low-level task only modifies the weights of the shared BiLSTM layer. With that implemented we achieve a full sterile BiLSTM layer for representation of POS tagging, and we have another representation as an input, which is the combination of the low-level training and the high-level training. The rest of the implementation is the same as our baseline BiLSTM (See 2.2).

2.3 Multiple Features MTL BiLSTMs

Next, we present our ultimate model, which is an MTL architecture that is able to store information about all the available morphological features. The idea is that a single feature MTL architecture might not capture a full representation of the token's morphology information that is essential for POS tagging. For different POS tags there exists different possible combinations of morphological features.

Taking the English language for example, a token with a correspondent tag 'Noun' may not have a morphological feature such as 'Tense', thus, while selecting 'Tense' as our single feature for MTL prediction, we narrow down our low-level representations for tokens of tag 'Noun'.

Moreover, tokens tagged with POS tags such as 'Pronoun' may have a rich morphological information that would not have been fully utilized had we restricted our architecture to only one low-level task.

The Multiple Features MTL (MFMTL) architecture works similarly to the Single Feature MTL (SFMTL) when observing a training procedure of a certain feature. However, the shared layer is now a single representation of various low-level tasks,

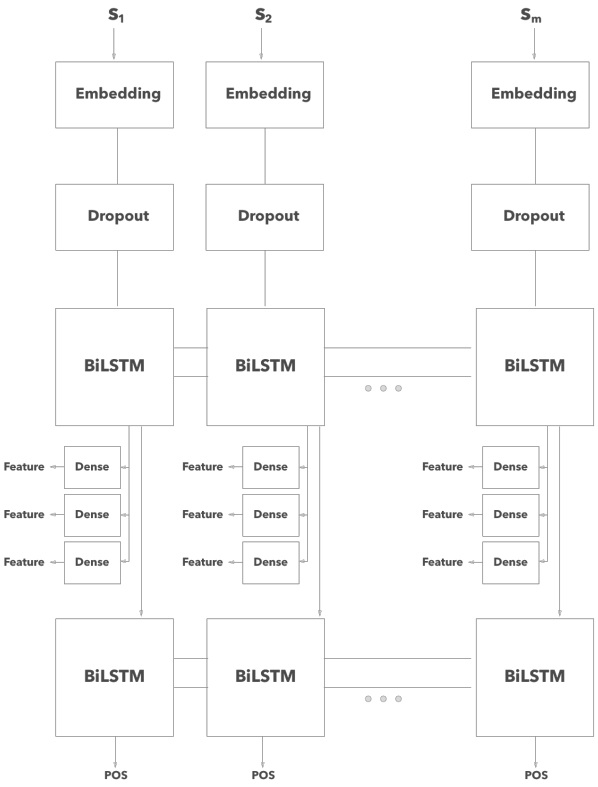

Figure 2: MTL with one multiple tasks

that is, during the training procedure, we feed the same BiLSTM shared layer regardless to which low-level task is currently observed.

What differentiates each low-level task during the training process is a unique dense layer that is adjusted separately depending on the low-level task that is being observed (See Figure 2). Formally, in the MFMTL model, for a token s_i we produce $n+1$ probability vectors: $\hat{f}_i^1, \ldots, \hat{f}_i^n, \hat{y}_i$ where n is the number of morphological features available in the language L. All other structural implementation is exactly as in SFMTL (See 2.2).

3 Experiments

3.1 Datasets

A major part of this paper is to examine whether the MTL method is valuable for better generalization of POS tagging. For such purpose, we evaluated our data from Universal Dependencies in two different methodologies.

Internal dataset evaluation The dataset was annotated as a whole and then divided into training, development and test sets.

External dataset evaluation Training, development and test sets were united into one dataset, used solely for training, and then other datasets

Language	Training #	Test #
English-EWT	12,543	4,079
Japanese-GSD	7,164	1,068
French-GSD	14,554	1,894
Portuguese-GSD	9664	2,414
Italian-ISDT	13,121	1,046

Table 1: Internal datasets evaluation no. of sentences

Language	Training Set	Test Set
English	EWT (16,622)	GUM (4,390)
Japanese	GSD (8,232)	PUD (1,000)
French	GSD (16,448)	ParTUT (1,020)
Portuguese	GSD (12,078)	Bosque (9,366)
Italian	ISDT (14,167)	PoSTWITA (6,713)

Table 2: External datasets evaluation and no. of sentences

training, development and test sets united for testing. See Table 2 for the allocation of data into training and test. Such methodology allows us to better evaluate how generalized our model is over the language because the sentences originate from different sources, they were annotated by different annotators and in different time.

3.2 Implementation Details

If s is a sentence of length greater than m , we truncate the last tokens and use only the first m tokens $\hat{s} = s_1, \ldots, s_m$, Otherwise, if s is a sentence of size $\hat{m}$ lower than m we use the special token ω so the new input is $\hat{s} = s_1, \ldots, s_{\hat{m}}, \omega, \ldots, \omega$.

The parameter m was chosen to be the the 95 length percentile for each language. Pre-defining the sequence length is crucial for two main reasons: Firstly, it allows us to perform training using mini-batches in parallel, and secondly, truncating the sequence may not harm the performance because RNNs are known to lose information as a sequence grows longer.

We determine that a token is "unseen" if when evaluating on the test set is the first time the token is observed, or if the token appeared in the training set less times than a defined threshold. In these experiments that threshold is defined to be 2.

3.3 Evaluation Metrics

Our main metric is the accuracy (number of correctly tagged tokens divided by all tokens), and we used two different, more rigid, evaluation metrics in order to assess how generalized our model is:

Unseen Accuracy is a metric that examines how well the model performs on tokens that were never observed during the training procedure.

Ambiguous Accuracy is a metric that examines how well the model performs on tokens that were indeed observed, but with more than one POS tag.

3.4 Results

4 Discussion

It is evident that our suggestion to evaluate the data over two completely different datasets decreases the amount of overfitting, and even the baseline BiLSTM reports averagely around 5% decrease in accuracy of all evaluation metrics.

Our research results imply that there is still room for improvement in POS tagging by BiLSTM models, even for the normal accuracy, when tested on uncorrelated test datasets.

It is fair to say that the results are not statistically significant enough to declare that both morphological models (SFMTL, MFMTL) are either improving or reducing the baseline model's performance over the tested metrics. Theoretically, if our task is to only POS tag a sequence of tokens, it is suggested to use the baseline model over the more complex models since it has less parameters, achieves similar results and thus may may generalize the problem quite better on smaller datasets.

5 Conclusions

Our results show that testing a trained BiLSTM model on a data portion annotated from correlated corpus may lead to biased results. Thus, our statement is that it is always more eligible to test the model on separated, sterile data. The above also suggests that the currently known state-of-the-art results in NLP should be re-evaluated in such manner if the data is available.

Multi-task learning models do not seem to strongly improve the baseline models when the low-level task is predicting token's morphological features, however, we do suggest that the same discipline should be examined on other low-level tasks to improve POS reported state-of-the-art results.

Acknowledgements

We would like to thank Dr. Jonathan Berant for teaching the NLP course at Tel-Aviv University and Dr. Kfir Bar for wise comments along the way. We also appreciate the hard work done in the multilingual annotated data in Universal Dependencies dataset.

References

Thorsten Brants. 2000. *TnT - A Statistical Part-of-Speech Tagger*. Saarland University, Germany

Dan Klein and Christopher D. Manning. 2003. *Accurate Unlexicalized Parsing*. pp. 423-430. Sapporo, Japan

Jess Gimnez and Llus Mrquez 2004. *SVMTool: A general POS tagger generator based on Support Vector Machines*. Lisbon, Portugal

Barbara Plank, Anders Søgaard and Yoav Goldberg 2016. *Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss*. Computing Research Repository

Anders Søgaard and Yoav Goldberg 2016. *Deep multi-task learning with low level tasks supervised at lower layers*. Association for Computational Linguistics

Sepp Hochreiter and Juergen Schmidhuber 1997. *Long short-term memory*. Neural Computation, 9:17351780.

M. Schuster and Kuldip K. Paliwal 1997. *Bidirectional recurrent neural networks*. IEEE Transactions on Signal Processing, 45(11):26732681, November.

Sebastian Ruder 2017. *An Overview of Multi-Task Learning in Deep Neural Networks*. Computing Research Repository, August

Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean 2013. *Efficient Estimation of Word Representations in Vector Space*. Computing Research Repository, August

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. 2014. *Dropout: a simple way to prevent neural networks from overfitting*. Journal of Machine Learning Research, 15:19291958