Part-of-Speech Tagging Comparison between Bidirectional LSTM-CNN and Bigram HMM for English and Chinese

Zhehan Shi
New York University
zs1113@nyu.edu

Abstract

State-of-the-art labeling system often requires engineering features manually, other systems offer an alternative to avoid the need to fine-tune features. This paper presents a comparison between two systems, a bigram Hidden Markov Model (HMM) and a bidirectional Long Short Term Memory (bi-LSTM) with character-level Convolutional Neural Network (CNN) and for English and Chinese. I build and evaluate two system for part-of-speech (POS) tagging task using two data sets, Penn Treebank WSJ corpus and Penn Chinese Treebank Corpus. The bigram HMM model achieves an accuracy for English and Chinese, 95.77% and 86.72% respectively, and the bi-LSTM-CNN model achieves an accuracy for English and Chinese, 96.84% and 91.51% respectively.

1 Introduction

In natural language processing (NLP), part-of-speech tagging is the process of assigning each word in a sentence of the corpus a corresponding part-of-speech tag (e.g., DT (Determiner) or NN (normal noun)), based on both the context and definition of the word. POS tagging is challenging because words can possess many dissimilar meanings in different context. It is indispensable to natural language processing because many other language-processing tasks, like syntactic parsing, can benefit from accurate tagging results.

When researchers and practitioners want to use a POS tagger, they would preferably choose an off-the-shelf POS tagger that is optimized for a particular language. If a suitable tagger is not accessible, two choices remain: a) using an existing tagger and pray that the result will be accurate enough, or b) implementing your own tagger, which sometimes requires specific knowledge. In a ideal situation, researchers and practitioners should be able to access to a language-independent tagger (Yasunaga et al., 2018).

Recent studies by (Plank et al., 2016) and (Horsmann and Zesch, 2017) evaluated POS tagger and compared the results to its HMM counterpart on corpora of numerous languages. Their evaluations conclude that the LSTM tagger reaches better results than those of HMM tagger. However, their studies do not include a comparison on Chinese. It is also worthwhile to explore if their results are reproducible on Chinese.

2 Hidden Markov Model

Hidden Markov Model (HMM) tagger for part-of-speech is both simple and effective, and is considered a case of Bayesian Inference. We observe a sequence of words and need to assign them the most probable sequence of POS tags. It relies on two major assumptions:

1. If the probability of a POS tag is only dependent on its previous POS tag

$$p(t_1^n) \approx \prod_i^n p(t_i|t_{i-1})$$

2. If the probability of a word is only dependent on its own POS tag

$$p(t_1^n|w_1^n) \approx \prod_i^n p(w_i|t_i)$$

If w_1^i denotes the tag sequence $w_i, \cdots, w_i$, and t_1^i denotes the tag sequence $t_i, \cdots, t_i$, given the assumptions for a bigram HMM tagger, the best tag sequence $\tau(w_1^n)$ for w_1^n can be calculated as:

$$\tau(w_1^n) = \text{argmax}_{t_1^n}\, p(t_1^n|w_1^n)$$

$$\approx \text{argmax}_{t_1^n} \prod_i^n p(t_i|t_{i-1})\, p(w_i|t_i)$$

with a set of transition parameters $p(t_i|t_{i-1})$, representing the probability of a tag occurring given the previous tag, and a set of emission parameters $p(w_i|t_i)$, represent the probability of a word occurring given the current tag.

$$p(w_i|t_i) = \frac{\text{count}(t_i, w_i)}{\text{count}(t_i)}$$

Training a simple bigram HMM tagger requires obtaining the counts from labeled data and normalizing to obtain the conditional likelihood. Viterbi algorithm is used as a standard approach for HMM decoding.

3 Neural Network Architecture

A character-level CNN takes character embeddings of the characters of each word to generate a vector representation of the word. The CNN word representation is then concatenated with a separate word embedding to create the input word representation of each word. A bi-directional LSTM then generates the output word vector representation of each word, which is fed through a linear projection and the softmax function, creating a likelihood distribution over the different POS tags of a word. CNN has been demonstrated by the previous studies (Ma and Hovy, 2016; Chiu and Nichols, 2016; Chen and Manning, 2014), to be an efficacious method to obtain morphological information from characters of words.

3.1 Word representation

Each word w_i is represented by an embedding vector $\mathrm{e}_i = \mathrm{E}_w[idx(w_i)] \in \mathbf{R}^{d_{emb}}$, acquired from a an embedding matrix, $\mathrm{E}_w[idx(w_i)] \in \mathbf{R}^{|V| \times d_{emb}}$, and d_{emb} is the word vector dimension, V is the vocabulary, and the operator $idx(w)$ maps a word $w \in V$ to a unique integer index, which corresponds to the row in E_w that contains the embedding vector for w.

Furthermore, the word also holds the representation of its character sequence. Each character $c_{i,j}$ of word w_i is represented by a character embedding $Y_{i,j} = \mathrm{E}_c[\text{char idx}(c_{i,j})] \in \mathbf{R}^{d_{char}}$, which is also acquired from an embedding matrix $\mathbf{R}^{d_{char}}$,where d_{char} denotes the character embedding dimension size, C is the set of characters, and $\text{char idx}(c)$ maps a character $c \in C$ to a unique integer index corresponding to a row in E_c.

These character embeddings are fed into a CNN, which has a 1D convolution that moves a sliding window of size k over the character sequence, using m different convolutional filters to each window in the character sequence. Max-pooling is utilized in the CNN.

3.2 Character-level CNN

As for the input to the CNN, $\overline{Y}_{i,j}$ is the concatenation of the character embeddings of the character and its $(k-1)$ neighboring characters:

$$\begin{aligned}\overline{Y}_{i,j} &= \left[Y_{i,j-(k-1)/2}; \ldots; Y_{i,j+(k-1)/2}\right] \\ &= \oplus\left(Y_{i,j-(k-1)/2\,:\,j+(k-1)/2}\right) \in \mathbf{R}^{k \cdot d_{char}}\end{aligned}$$

The m different convolution filters, $u_1, \cdots u_m$ are arranged into a matrix U and with a bias vector b_{conv}, and the convolution operation is defined:

$$\phi_{i,j} = g(\mathrm{b}_{conv} + \mathrm{U} \cdot \overline{Y}_{i,j})$$

$$\mathrm{b}_{conv} \in R^m; \phi_{i,j} \in R^m; \mathrm{U} \in \mathbf{R}^{k \cdot d_{char} \times m}; \overline{Y}_{i,j} \in \mathbf{R}^{k \cdot d_{char}}$$

Through the character convolution for word w_i, there are vectors $\phi_{i,j} \cdots \phi_{i,|w_i|}$. These vectors are pooled into a single vector, $\mathrm{q}_i \in R^m$, representing the character sequence of the word w_i. Using max-pooling operation, the value of x-th dimension of the vectors $\phi_{i,j} \cdots \phi_{i,|w_i|}$

$$\mathrm{q}_i[x] = \max_{1 \le j \le |w_i|} \phi_{i,j}[x] \quad \forall x \in \{1, \cdots, m\}$$

A word w_i is represented by the character-level representation q_i and its embedding vector e_i. The input representation of w_i is

$$\hat{\mathrm{w}}_\mathrm{i} = [\mathrm{q}_\mathrm{i}\,; \mathrm{e}_\mathrm{i}]$$

3.3 bidirectional LSTM

The word representation vectors $\hat{\mathrm{w}}_\mathrm{i}$ are fed sequentially to the forward LSTM from left to right and to the backward LSTM from right to left. For each time step i corresponding to the position of a word in the sentence, the forward LSTM generates a hidden representation vector $\overrightarrow{\mathrm{h}_i} \in \mathbf{R}^{d_h}$, which is fed to the next time step, while the backward LSTM produces $\overleftarrow{\mathrm{h}_i} \in \mathbf{R}^{d_h}$ which is fed to the previous time step, where d_n represents the dimension of the LSTM hidden representation vector.

The hidden representation of each word w_i concatenation of both forward and backward LSTM hidden representation vectors

$$\mathrm{h} = [\overrightarrow{\mathrm{h}_i}\,; \overleftarrow{\mathrm{h}_i}] \in \mathbf{R}^{2d_h}$$

The hidden vector of the LSTM from $2d_h$ dimensions will be projected through linear projection to the corresponding dimension, which is size of the tagset.

$$\mathrm{s}_i = \mathrm{W}_t \cdot \mathrm{h}_i + \mathrm{b}_t$$

where d_t is the size of dimension that corresponds to the tagset size, $\mathrm{s}_i, \mathrm{b}_i \in R^{d_t}$ and $\mathrm{W}_t \in R^{2d_h \times d_t}$.

The likelihood of a POS tag t being the tag at position i, namely $t_i = t$, is calculated as

$$\mathrm{p}(t_i = t | w_1, \cdots, w_N) = \frac{\exp(\mathrm{s}_i[\mathrm{tag\,idx}(t)])}{\sum_{a=1}^{d_t} \exp(\mathrm{s}_i[a])}$$

where the function $\mathrm{tag\,idx}(t)$ maps a POS tag t to an integer index in $\{1, \cdots, d_t\}$.

4 Evaluation Corpora Dataset

4.1 English

For POS tagging task in English, I select sections 02-21 from the Penn Treebank WSJ Corpus for the training set, which contains 950028 tags and 39832 sentences. The development set is from section 24, containing 32853 tags and 1346 sentences. The test set is from section 23, containing 56684 tags and 2416 sentences.

	TRAIN	**DEV.**	**TEST**
TAGS	955028	32853	56684
SENT.	39832	1346	2416

4.2 Chinese

For POS tagging in Chinese, I use a significant portion of Penn Chinese Treebank (LDC2007T36). I pre-processed the data so that it has the same format as the WSJ corpus, removing redundant HTML tags. The training set contains 508613 tags and 18784 sentences. The development set contains 21557 tags and 659 sentences. The test set contains 23626 and 786 sentences.

	TRAIN	**DEV.**	**TEST**
TAGS	508613	21557	23626
SENT.	18784	659	786

5 Results

The bigram HMM model is used for the baseline model. On Chinese, the baseline model achieves a development accuracy of 87.44% and a test accuracy of 86.72%. On English, the baseline model achieves a development accuracy of 95.32% and a test accuracy of 95.77%. These results are used as baseline accuracy.

The bi-LSTM-CNN model achieves a development accuracy of 96.58% and a test accuracy of 96.84% on Chinese. On English, the new model achieves a development accuracy of 96.58% and a test accuracy of 96.84%.

5.1 English

	DEV. ACC.	**TEST ACC.**
BASELINE	95.32%	95.77%
bi-LSTM-CNN	96.58%	96.84%

5.2 Chinese

	DEV. ACC.	**TEST ACC.**
BASELINE	87.44%	86.72%
bi-LSTM-CNN	91.50%	91.51%

6 Conclusion

The paper evaluated the performance of bidirectional LSTM tagger using character-level convolutional neural network and that of the bigram HMM tagger with features engineering across 2 languages. The results are in accordance with the findings (Horsmann and Zesch, 2017; Yasunaga et al., 2017; Plank et al., 2016) of other researchers. The LSTM tagger achieves a superior result regardless the language involved, truly remarkable as Chinese does not employ any letter of the alphabet. For English, it increase more than 1% from 95.77% to 96.84%. For Chinese, it gains more than 4% from 86.72% to 91.51%. The downside from the LSTM approach is the time taken to train the model. It takes much longer time to train LSTM model than that of the HMM model; however, the time taken to train model cannot be estimated properly since HMM approach requires a significant time commitment in designing the appropriate features to boost the overall accuracy. Subtoken representations are indispensable in avoid engineering features manually.

It is safe to surmise that near-term improvement on sequence tagging, like POS tagging, rests upon upgrading LSTM model with other steps, including but not limited to Conditional Random Field and attention layer.

References

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Tobias Horsmann and Torsten Zesch. 2017. Do LSTMs really work so well for PoS tagging? – a replication study. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 727–736, Copenhagen, Denmark. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Michihiro Yasunaga, Jungo Kasai, and Dragomir Radev. 2018. Robust multilingual part-of-speech tagging via adversarial training. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 976–986, New Orleans, Louisiana. Association for Computational Linguistics.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 452–462, Vancouver, Canada. Association for Computational Linguistics.