Transition-Based Dependency Parsing

YWJ
HUST-IDC / No Address
ywj@hust.edu.cn

Abstract

I'm going to give a quick review on parsing task and particular introduction of a specific LSTM-based parser (Dyer et al., 2015) in this article. The main content was arranged as below: (i) A brief scan on Transition-based method and dependency parsing. (ii) An introduction of the certain parser . (iii) A particular introduction of my implementation. (iiii) A short discussion on the parser. Visit this link for the code.

1 Introduction

Followers of rule-based methods believe a sentence valid in natural language must be restricted by latent rules. A obvious rule would be easily summarized by every natural language speaker is that structure of any language is recursive, namely complexer sentence always be made up by shorter phrase recursively. Directly, tree structures which is also recursive has been used to modelling sentences. When the tree correspond to sentence was built following certain rules, result looks hopefully since the main word of a sentence always gather to the top of tree. We attach more importance to some parts of a sentence than the others, because the former are believed more relevant to the sentence's semantic. For example, "A fat cat jump over the lazy dog" actually means "cat jump dog". Although the latter is unidiomatic one, it still represents most meaning of the former, for the former is about jumping but not whether cat was fat or dog was lazy. Well defined parsing rules should find out the point in any sentence, for the example above, parser should give a result like "[A [fat [cat]]] [[jump] over] [the [lazy [dog]]]", but unlike anything else. Many works (Li et al., 2015; Bowman et al., 2014; Tai et al., 2015) have given the same view that semantic sensitive task benefit from tree structure in neural network-based method.

There are many standards to parse a sentence and all of them can be divided into two categories, namely constituent parsing and dependency parsing. Four axioms was proposed to restrict the procedure of generating dependency parsing tree (Robinson, 1970): (i) There is only one single independent constituent in one sentence. (ii) All the other constituents of the sentence must dependent on certain constituent. (iii) Any constituent won't dependent on two or more constituent. (iiii) If A dependent on B, and C located between A and B in the sentence, then C must dependent on A or B or a constituent between A and B. Note that following the axioms is necessary but not sufficient condition for a valid dependency parsing tree. Actually, there is no specific rules to indentify whether a parsing tree is valid or not. Instead, for any parsing standard, a artificial data set is given to measure the result of a certain parser, so called gold-standard data set.

Transition-based dependency parsing formalizes the procedure of generating a parsing tree as a series of decisions while reading a sentence sequentially. Arc-standard (Nivre, 2004) is a widely used transition inventory. Arc-standard transitions parse a sentence from left to right, using a stack to store partially built syntactic structures and a buffer that keeps the incoming tokens to be parsed. On each step parser choose an action from a set according to the decision given by a learner. Arc-standard give a set containing four actions: {SHIFT, REDUCE, LEFT-ARC, RIGHT-ARC}. But in Dyer's parser, a simplified but equivalent set have been choosen: {SHIFT, LEFT-REDUCE, RIGHT-REDUCE}. The meaning of each action shows in Table 1.

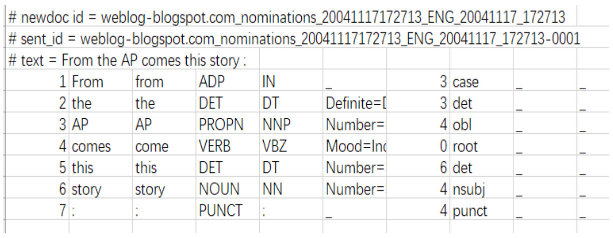
Stack$_t$	Buffer$_t$	Action	Stack$_{t+1}$	Buffer$_{t+1}$	Dependency
S,u,v	*B*	RIGHT-REDUCE	*S,u*	*B*	$u \rightarrow v$
S,u,v	*B*	LEFT-REDUCE	*S,v*	*B*	$v \rightarrow u$
S,u	*v,B*	SHIFT	*S,u,v*	*B*	———

Table 1: Arc-standard action in Dyer's parser

2 Transition-Based Dependency Parser with LSTM

2.1 CoNLL Dataset

CoNLL competition is annual meeting focus on semantic tasks. CoNLL dataset (Nivre et al., 2017) including multilingual tree-banks produced by gold-standard stanford parser. All data have been written in conllu format, which including information of POS tag and parsing tree, see Figure 1. Arborator (Gerdes, 2013) is a open source visualize tool for conllu files. It offers a online window for quick viewing conllu files.



```
# newdoc id = weblog-blogspot.com_nominations_20041117172713_ENG_20041117_172713
# sent_id = weblog-blogspot.com_nominations_20041117172713_ENG_20041117_172713-0001
# text = From the AP comes this story :
      1 From    from    ADP     IN     _            3 case    _    _
      2 the     the     DET     DT     Definite=[   3 det     _    _
      3 AP      AP      PROPN   NNP    Number=      4 obl     _    _
      4 comes   come    VERB    VBZ    Mood=Inc     0 root    _    _
      5 this    this    DET     DT     Number=      6 det     _    _
      6 story   story   NOUN    NN     Number=      4 nsubj   _    _
      7 :       :       PUNCT   :      _            4 punct   _    _
```

Figure 1: Sentence "From the AP comes this story" in conllu format.

2.2 Dyer's Parser

Follow the former state-of-art parsers (Vinyals et al., 2014; Chen and Manning, 2014), Dyer proposed a parsing model uses three stack LSTMs: one representing the stack sequence, one representing the buffer sequence, and one representing the history of parse actions to encode parser states. In Figure 2, S designates the stack of partially constructed dependency subtrees and its LSTM encoding; B is the buffer of words remaining to be processed and its LSTM encoding; and A is the stack representing the history of actions taken by the parser. These are linearly transformed, passed through a ReLU nonlinearity to produce the parser state embedding P$_t$. An affine transformation of this embedding is passed to a softmax layer to give a distribution over parsing decisions that can be taken.

2.3 Measurement Standard of Parser

Label attachment score (LAS) and unlabeled attachment score (UAS) are the most important criterion for parser. Other indexes such as label accuracy (LA) sentences per minute (SPM) also make sense but not the primary. When evaluating, all tokens including punctuations are taken into account.

3 Implementation

3.1 Transform Parsing Tree into Transition Sequence

Arc-standard defines the rules building a parsing tree according to transition sequence. For training, transition sequence should be deduced reversely from the parsing tree writing in conllu format. To tackle this, a naive algorithm was given below, it takes $O(\frac{n^2+3n}{2})$ to handle a parsing tree with n words at worst case.

Algorithm 1 Tree2Transition

```
Input: S=sentence  Output: T=transition
 1: stack.push(root)
 2: for word in S do :
 3:     S.pop(word)
 4:     stack.push(word)
 5:     T.add(SHIFT)
 6:     while True do :
 7:         if stack(top-1)→stack(top) && IsIndependent(stack[top]) then :
 8:             stack.pop(top)
 9:             T.add(RIGHT-REDUCE)
10:         else if stack(top)→stack(top-1) && IsIndependent(stack[top-1]) then :
11:             stack.pop(top-1)
12:             T.add(LEFT-REDUCE)
13:         else: break
14: return T
15: function IsIndependent(word, S, stack)
16:     for word* in S, stack do :
17:         if word→word* then :
18:             return False
19:     return True
```

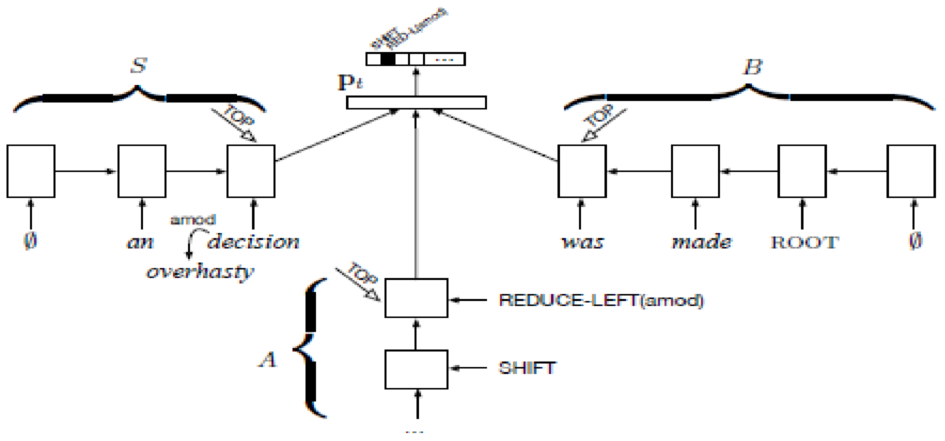


Figure 2: Parser state computation encountered while parsing the sentence "an overhasty decision was made".

3.2 Generate Sample List from Conllu Files

I defined class conll to deal with files in conllu format. Since the sequence encoded by LSTM is changeable due to transition actions, sentence was pre-transform into several versions equal to number of actions in transition sequence. That means at each step of generating parsing tree, class conll would take a picture of stack, buffer and transition history, then pack them together and throw to the sample list. Consider variety length of sequence, all samples in a batch was align to the longest sequence among them, the shorter ones were padded with zero-vector in the front of the sequence. A little difference made here is, I use a floating batch size to ensure that samples in each batch come from the same sentence.

3.3 Pre-trained Embeddings

I trained word embeddings with dimension of 300 on Wikipedia-corpus, which has a dictionary of one million tokens. For simplicity, I used Gensim (Řehůřek and Sojka, 2010) to finish the training. Gensim is a open source tools package for probabilistic graphical model, it also provides tools for word embeddings' training support multiprocess on cpu and convenient binary storage for dictionary and embedding matrix.

For part of speech (POS) embeddings, practice showed that one-hot representation for POS tag doesn't really works well here. I trained POS tag embeddings with dimension of 20 on the corpus abstract from conll tree-bank. As transition actions, I add one more fully connect layer before softmax to embed 58 kinds of transition. Note that except transition embeddings were treat as weight of network and would be updated during training, all the other embeddings were fixed during training.

3.4 Loss Computation

To implement object on maximum likelihood, a mask was generate according to labels so that only the predict probability of correct transition would contribute to loss. Loss take the mean of all probabilities' logarithm. To apply gradient descent, final loss multiplied by -1 to ensure minimum existent.

To visualize the training process, I give a over-simplified loss estimation here. When initiate, we can treat all probability given by softmax are ideally equivalent. Since there are 58 different transition actions, the initiate loss could be evaluated as below:

$$LOSS_{init} = -\ln(\frac{1}{58}) \approx 4.06$$

In fact, I have only launched the training procedure twice and the initiate loss were 4.4 and 4.6 according to my unfaithful memory. When training finished, loss stabilized around 0.2, which means the mean of predict probabilities are around:

$$MEAN_{pro} = e^{-0.2} \approx 0.81$$

3.5 Discussion

An obviously shortcome of transition-based method is that the prediction of latter transition rely on correct prediction of all transition in front of which. This lead to the problem that once a wrong prediction was given, all the predictions behind it went wrong. Another ambiguous point is the measurement of parser. As noted in the previous article, both LAS and UAS evaluate the correct ratio of tokens in range of whole tree-bank. But in my opinion, a parsing tree is bad one even though a single token was predicted wrongly inside. And no matter how many wrongly predictions were given in a sentence, we actually got one wrongly parsed sentence. That means evaluating a parser by the correct ratio of sentence parsing is the better way, to prevent a highly evaluated parser rarely giving a wholly correct parsing tree.

References

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2014. Recursive Neural Networks Can Learn Logical Semantics. *arXiv:1406.1827 [cs]* ArXiv: 1406.1827. http://arxiv.org/abs/1406.1827.

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. http://www.aclweb.org/anthology/D14-1082.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. *arXiv:1505.08075 [cs]* ArXiv: 1505.08075. http://arxiv.org/abs/1505.08075.

Kim Gerdes. 2013. Collaborative Dependency Annotation. In *Dependency Linguistics*. Prague, Czech Republic, Proceedings of Dependency Linguistics, page 88. https://halshs.archives-ouvertes.fr/halshs-01509073.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185* https://arxiv.org/abs/1503.00185.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. Association for Computational Linguistics, pages 50–57. http://dl.acm.org/citation.cfm?id=1613156.

Joakim Nivre,
v Zeljko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, and Bick. 2017. Universal dependencies 2.0 LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (
'UFAL), Faculty of Mathematics and Physics, Charles University. http://hdl.handle.net/11234/1-1983.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, pages 45–50. `http://is.muni.cz/publication/884893/en`.

Jane J. Robinson. 1970. Dependency structures and transformational rules. *Language* 46(2):259–285.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *arXiv:1503.00075 [cs]* ArXiv: 1503.00075. http://arxiv.org/abs/1503.00075.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2014. Grammar as a Foreign Language. *arXiv:1412.7449 [cs, stat]* ArXiv: 1412.7449. http://arxiv.org/abs/1412.7449.