naveen.md

NLP ASSIGNMENT #3

```
Naveen Singh Pundir
17111026
CS671A: Introduction to Natural Language Processing
IIT KANPUR
```

This file contains the description of techniques used in the code for assignment3.

To run the python code, make sure you have dataset present inside the `data` folder in the parent directory.

Directory Tree:

```
Assignment1
    ├── utils
    │   └── init.py
    │   └── feature_extraction.py
    │   └── general_utils.py
    └── data
    │   └── dump
    └── model.py
```

To run the code:

```
$ python model.py
```

Requirements

Before running the scripts make sure that you have following dependencies installed in your system.

- Python 3.6
- Numpy
- Matplotlib
- NLTK
- Keras
- Tensorflow

Description

High level idea

The ideas is taken from word2vec to create dense feature vectors for words, pos and dependency relations. The neural network has 4 layers with two hidden layer of size 200 and 15 . Hidden layer values are computed by cubic activation function(Cubic activation allows interaction between all the all three parts and between all possible pairs) .

Input

Input made up of word2vec vectors of address words, corresponding
vectors for PoS tags, label vectors for the tree addresses chosen
from the embedding matrices.

Features

Following (Zhang and Nivre, 2011), we pick a rich set of elements for our final parser. In detail, S^w contains $n_w = 18$ elements:

(1) The top 3 words on the stack and buffer: s_1, s_2, s_3, b_1, b_2, b_3;

(2) The first and second leftmost / rightmost children of the top two words on the stack: $lc_1(s_i)$, $rc_1(s_i)$, $lc_2(s_i)$, $rc_2(s_i)$, i = 1, 2.

(3) The leftmost of leftmost / rightmost of rightmost children of the top two words on the stack: $lc_1(lc_1(s_i))$, $rc_1(rc_1(s_i))$, i = 1, 2.

We use the corresponding POS tags for S^t (n_t = 18), and the corresponding arc labels of words excluding those 6 words on the stack/buffer for S_l (n_l = 12). A good advantage of our parser is that we can add a rich set of elements cheaply, instead of hand-crafting many more indicator features.

Loss Function

We have used categorical entropy loss function.

Prediction

To predict the transition operation an input vector is created from the embedding matrices for the configuration c_i and then fed forward through the learnt network to predict the transition t_i at the softmax layer. The next configuration is obtained by $c_{i+1} = t_i(c_i)$.

Training Dataset

http://universaldependencies.org/ (use the EWT treebank for English)

Accuracy in Test Set

Accuracy = 84.65%

Conclusion

Our model only relies on dense features, and is able to automatically learn the most useful feature conjunctions for making predictions. An interesting line of future work is to combine our neural network based classifier with searchbased models to further improve accuracy. Also, there is still room for improvement in our architecture, such as better capturing word conjunctions, or adding richer features (e.g., distance, valency).