Decoding a Generalized Neural Architecture for assigning labels to words (This is a school project and not for ACL submission!!!)

Anonymous ACL submission

Abstract

This project is about a generalized neural network architecture that could be used for assigning labels to words in NLP tasks. The proposed system will be able to assign NER labels, POS Tags and can chunk sentences given a small training set.This implementation is based on the paper by May and Hovy ,2016.

1 Introduction

Most of the state of art systems currently in practice are specifically tailored for a single task with hand engineered features or neural networks specific for one task. Collobert et al (2011) called for and proposed a centralized architecture. Having a centralized architecture saves on development and debugging cost. There have been several improvements on this architecture.

2 Goals

The Goals of this project are two fold :

1. Decrease the run time of the implementation the authors have in theano / lasagne. The library of choice is tensorflow

2. provide insights on what each layer in the model does thereby increasing the interpretability of the model

This current project is a slight variation of one such papers (Ma & Hovy , 2016) which came up with a LSTM-CNN-CRF based system. The authors used a Lasagne based code and one the contributions of this paper is a tensorflow based implementation of the similar system. Preliminary studies of both implementation on a same system shows the tensor-flow implementation to be 2 times faster than the published theano version. Another contribution of this project is the explanation it provides in understanding the components of the model. The model will be explained in terms of

1. why it assigned a label to the word in simple terms

2. understanding the role of CNN used for Character embedding

3. Understand the role of the word embedding layer

3 Prior work

There has been a couple of papers in the recent past using varying combinations of CNN or RNN with LSTM and CRF. One of the earliest papers in this area is from Baidu (Zhou & Xu. 2015) which is builds on Collobert et al.,2011 where they discuss a NN architecture with word em-bedding, CNN and CRF layer. The word em-beddings address the data sparsity problem while CRF is commonly used method in su-pervised slot tagging applications. However, long range dependencies are not modelled sufficiently by CNN which only includes words within a limited context. Some research groups (Melamud et. al, 2015 and Levy & Goldberg,2015) attempted to solve long range dependency problem by using a dependency parser. The authors from Baidu propose using a deep BiLSTM (i.e. 2 layers of BiLSTM) followed by a CRF. LSTM was selected because it can efficiently handle vanishing/exploding gradient while allowing words further apart in the sentence to influence the current word. The research groups which used dependency parser later published an-other paper where they created context2vec (Melamud et. al., 2016). context2vec has a BiLSTM followed by a MLP (multi-layer perceptron). Context2Vec helps

identify words that fit in a similar context. Another interesting work is a paper from CMU (Ma I& Hovy, 2016). The authors de-scribe an end-to-end system without need for handcrafted feature in detail.

4 Task Description & Review of existing literature

Here is a high-level summary of the tasks involved with a focus on named entity recognition. While considering entity tagging problems, here are a couple of intuitions that will help us in building a good language model.

1. Morphological information: (like the prefix or suffix of word) provides pre-liminary hint on the type of entity a word is. Ma and Hovy (2016) used CNN over characters of a word to model this.

2. Names often span across words so modeling must be done jointly across multiple tokens (eg. Substrate Intelligence). Linear CRF is often a method of choice (Ma and Hovy,2016; Lample et al., 2016; Zhou & Xu, 2015). It uses the transition probability (i.e. probability of I-ORG given B-ORG) and emission probability (i.e probability of the word given the tag I-ORG) to generate output tags with maximum likelihood. By using LC-CRF, NER grammar rules like I-ORG cannot follow I-PER or I-PER cannot follow B-LOC are implicitly encoded in the model

3. Orthographic evidence: This refers to the internal characteristic of something that looks like a name. For example, in some of our project names (most likely abbreviations) like QAS or LUNA we have u following L and a following Q. In non-personal words, Q is almost often followed by u. Lample et al (2016) modeled this by using Bi-LSTM at the character level. This will help us in modeling words which are outside our known vocabulary even when we have very little context for them.

4. Distributional evidence: As Firth mentioned way back in 1950s, a word is known by the context it keeps. Over the last year, most p-pers (Melamud et al., 2016; Ma & Hovy,2016; Lample et al., 2016; Zhou & Xu, 2015) have used Bi-LSTM almost unanimously to address this issue. BiLSTM provides past and future context to a word.

5. Data sparsity: is another issue which has been addressed using pretrained embedding and by training new embeddings specific for the data in hand.

6. Other features like Capitalization can also be a good indicator of an entity.

However, not all these characteristics are always present in every sample. Some names can sound like regular words eg., keyphrase Identification. Some of our project names are fully in capitals while others may have just the first letter capitalized. So, many papers (Ma & Hovy,2016; Lample et al., 2016; Zhou & Xu, 2015) report using drop out layers to address this issue and to solve the problem of overfitting.

5 Methods

5.1 Networks Architecture

The high level system architecture can be summarized by the diagram from Ma I& Hovy, 2016.The method used is similar to the above paper with some minor variations. Dataset: ConLL 2003 NER dataset with 14041 / 3247 / 3450 sentences in train / dev / test.

Word Embedding: We read in 6 billion pretrained vectors from Wikipedia and web (open source).For every word in the vocabulary from training set, if it does not have a pretrained vector randomly initialize it.Word dimension size is set to 100

Character representations: We generate random embeddings for every character in the vocabulary. Embedding dimensions are set to 30. We add a drop out layer with p=0.5 to prevent over fitting.Character representations need to be converted to word format so we pass them into a Convolutional Neural Network (CNN) with a filter length =30 and window size of 3.Activation function is tanh.We use conv 1 D , activation as tanh, full padding

Merge Layer: Word Embeddings and Character representations are merged together and then selected with a dropout rate of 0.5

BiLSTM:Use the standard LSTM architecture as de-scribed by Hchreiter & Schmidhuber , 1997. LSTM hidden layer dimension are set at 200.A third drop out layer with p =0.5 is added

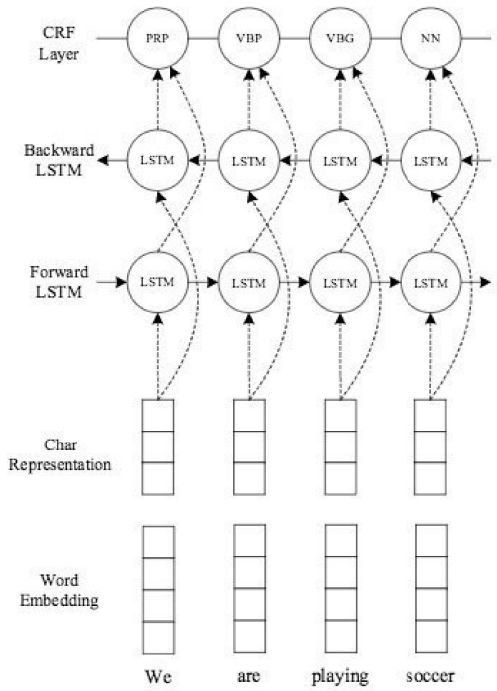

Figure 1: Model Architecture taken from Ma &Hovy, 2016.

5.2 Network shape

The network is composed of 3 layers. Each layer in terms of its output shape is described below(The input and out shape for each layer is described int he code):

- Layer 1 : This layer consists of character representations and word embeddings.
  - ⋆ Character Representation layer: [batch_size, max_char_per_word * sequence_length, char_embedd_dim]
  - ⋆ Word Embedding layer :[batch_size, sequence_length, word_embedd_dim]
- Layer 2 : Forward and Reverse LSTMs (BiLSTM) along with an output linear dense layer to squash the results
  - ⋆ Right LSTM : [batch_size, sequence_length, Hidden_Unit_Size]
  - ⋆ Left LSTM : [batch_size, sequence_length, Hidden_Unit_Size]
  - ⋆ BilSTM : [batch_size, sequence_length, 2 * Hidden_Unit_Size]
  - ⋆ Linear Layer : [batch_size, sequence_length]
- Layer 3 : CRF Layer
  - ⋆ CRF : [batch_size, sequence_length]

5.3 Experimental Variables

Following experimental variables were changed during the course of experimentation:

- Gradient Clipping : Initially, there was an exploding gradient issue with the network returning NaNs after few runs. I tried Global gradient clipping (absolute Vs Clip by Norm) , clipping after BiLSTM. I found absolute clipping after BiLSTM to give the best results. So it was used in subsequent experiments.
- Dropout: The paper talked about 3 dropout layers interspersed within the network. I tried adding an additional dropout layer after BiLSTM and the accuracy dropped by more than 10 points.
- Optimizer: The paper used SGD and mentioned that other optimizers gave similar results. I struggled with SGD for a long time and then just changed it Adam Optimizer. That small changes greatly increased the accuracy
- Batch Size: Good results were obtained with small batch size i.e between 10 and 64
- Filter Size and Number of Filters in CNN for character representation: Changing these values did not change the P/R numbers

5.4 Interpretability

The context which contributed to assigning particular label to a given word was obtained by modifying

- Ribeiro et. al., 2016 code to accommodate CRF function.
- Adapting the CRF Viterbi decode to output probability (like) scores.

The algorithm takes a sentence and creates variations by randomly removing words as shown in the Figure 2. Each of the variations is passed through the prediction function. The prediction function reads in the sentence and the target word and outputs the probability that the target word belongs to one of the 18 classes. We have 17 classes in the

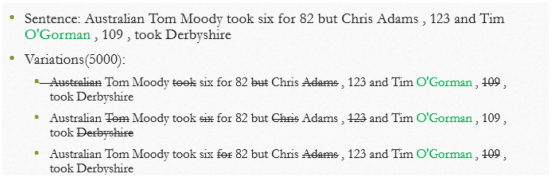

Figure 2: Variations of a sentence for predictions

training set and we add an extra unknown class when we created the model. Now, output from the predictor and the sentence is passed with the output as label to the ridge regressor. The ridge regressors assigns weights to each of the context words

6 Evaluation

Extrinsic evaluation we are interested in is the F1 score. The F1 score is calculated as described by Sang & Meulder , 2003. A named Entity is correct if it is an exact match of the true entity.

$$F_\beta = \frac{(\beta^2 + 1) * precision * recall}{\beta^2 * precision + recall} \quad (1)$$

The intrinsic loss that the neural network optimizes is the negative log likelihood of the CRF loss as described by Ma & Hovy,2016.

7 Results and Discussion

7.1 Tensorflow Implementation

The Tensorflow implementation of the archetecture takes about 8 hours as opposed to 36 hours for the theano implementation on the same machine. I tried using both Gradient descent and Adam Optimizers and found the Adam Optimizer works better than stochastic gradient descent. Also,I clipped the LSTM layer instead of the whole network. I tried changing other parameters like the learning rate and decay rate and found the values suggested in the paper to be most optimal. The whole implementation can be obtained from https://github.com/LopezGG/NN_NER_tensorFlow

7.1.1 Precision & Recall

The overall statistics on the test set with break down per category is

	ORG	LOC	MISC	PER	Other
Precision	57.72	74.41	62.69	69.82	97.19
Recall	52.18	77.39	63.89	66.94	98.04
F1 score	54.81	75.87	63.28	68.34	97.61

7.2 Interpretability

7.2.1 Character Embedding layer

Here are some of the words which were not in the training and development set vocabulary. These words are called Out of Unsupervised Vocabulary

9-0-54-1	disapproval	highest-ranking
9-0-49-1	disgraceful	biggest-ever
6-0-30-1	scrapped	Rochester
6-0-40-1	distracted	rapidly-growing
4-0-18-0	dishoarding	higher-than-expected
8-0-37-2	non-partisan	self-government

The character embedding layer groups words which share a similar pattern as shown above. This helps with categorizing unknown words

7.2.2 Word Embedding layer

The original pretrained word embeddings were grouped together by their semantic meaning. After running those pretrained words through this network , the Word embedding layer now is optimized to group words belonging to the same category as shown in the table below

Commission:E-ORG	Democratic:S-MISC
Assembly:E-ORG	GMT:S-MISC
Dortmund:E-ORG	English:S-MISC
University:E-ORG	Moslem:S-MISC
Ministry:E-ORG	Thai:S-MISC
TVM:S-ORG	Polish:S-MISC

It is to be noted that many of the words belong to multiple categories. for eg. the first word "Commission: E-ORG: 32; S-ORG: 11; B-MISC: 4; I-ORG: 2 " belongs to E-ORG category in 32 instances. The categories to which the word "Commission" belongs to depends on the context. The table above shows the most frequent class assigned to each word.

7.2.3 Debugging the CRF layer

The results for interpretting the CRF layers are as follows:

- Sentence: Australian Tom Moody took six for 82 but Chris Adams , 123 and Tim

O'Gorman , 109 , took Derbyshire The target word is "O'Gorman" with a tag E-PER which the system has never seen before in the development or training set. The model shows that the presence of "Tim" , B-PER had a high influence on assigning E-PER to "O'Gorman"

- Sentence: Anthony Hill (Australia) beat Dan Jenson (Australia) 15-9 15-8 Similarly, in this case the target word is Hill (E-PER). The model described above confirms that the presence of Anthony (B-PER) along with the presence of other 'O' category words like "beat" ,"(" and ")" contributed to assigning E-PER to Hill

- Sentence: West Indian all-rounder Phil Simmons took four for 38 on Friday as Leicestershire beat Somerset by an innings and 39 runs in two days to take over

  The target word in this case was "Indian" (E-MISC). the presence of "West" (B-MISC) contributed to this assignment.

The actual scores from the models for all these sentences are shown in figures

8 Challenges & Opportunities

We need a better way to visualize the results of LSTM and CRF. The current implementation is hacky because the CRF-decode does not actually output probability. I will have to read and explore other papers to make this calculation more robust

9 References

Felix A Gers, Jurgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. Neural computation, 12(10):24512471.

Jie Zhou and Wei Xu. 2015. End-to end learning of semantic role labeling using recurrent neural networks. In Proceedings of the Annual Meeting of the Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. Journal of Marchine Learning Research, 12:24932537, November

Tjong Kim Sang, Erik. F. 2002. Introduction to the CoNLL-2003 Shared Task: Language Independent Named Entity Recognition. In Proc. Conference on Natural Language Learning

Omer Levy and Yoav Goldberg. 2014. Dependency based word embeddings. In Proceedings of ACL.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In Proceedings of Workshop on Vector Space Modeling for NLP (VSM).

Oren Melamud, Jacob Goldberger, Ido Dagan. 2016. context2vec: Learning Generic Context Embedding with Bidirectional LSTM. CoNLL Xuezhe Ma, Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF arXiv:1603.01354

Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin. 2016 "Why Should I Trust You?": Explaining the Predictions of Any Classifier ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD),

10 Appendix

This project was done solely for "Learning to care for Neural Networks" course by Lopez Gilsinia Gilroy with help from Jeremy Kahn and some inputs from Marco Tulio Ribeiro for the interpretability part.