Named Entity Recognition with Deep Learning

Presentation by:
Reza Ghaeini
Farhana Ferdousi Liza

Oregon State University

Index

- Named Entities Recognition
- Deep Learning
- Word Representation
- Recurrent Neural Network
- Dataset
- Experimental Result
- Future works

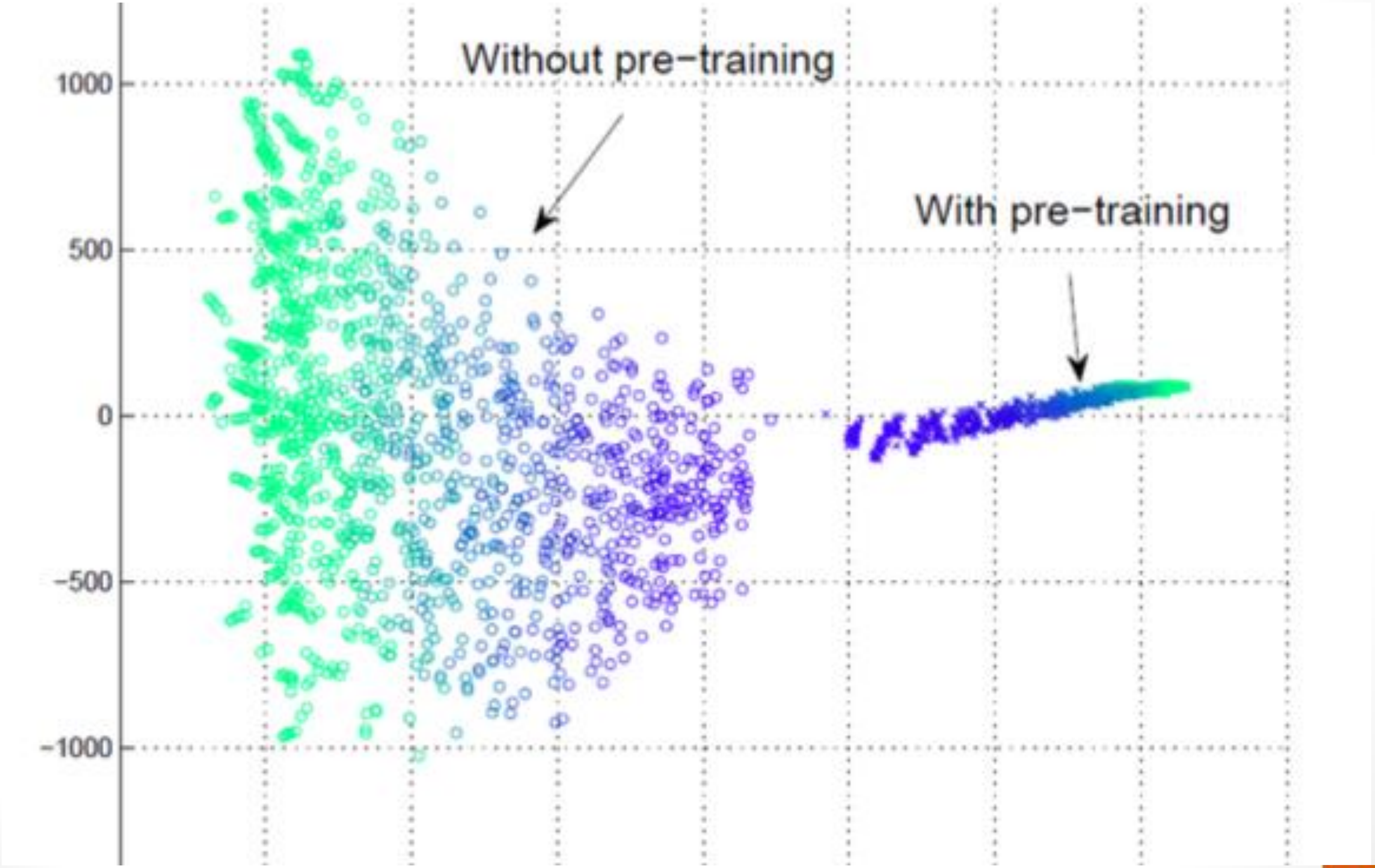
Named Entity Recognition

- Detecting named entities like name of people, locations, organization etc. in a sentence and text.
- Example:
  - Input:
    - Jim bought 300 shares of Acmpe Corp. in 2006.
  - Output:
    - [Jim]$_{Person}$ bought 300 shares of [Acmpe Crop.]$_{Organization}$ in [2006]$_{Time}$

Deep Learning

- Recently, it outperforms many tasks in different areas like NLP
- It tries to capture deep features of data by itself (no feature extraction …)
- It use pre-trained information (generally unsupervised) (i.e. word representation) to achieve better result

Deep Learning



Word Representation

- Each word in vocabulary associated with n-dimensional vector
- Capture similarity between words in different aspects
- It can capture interesting relations too.
- $[WR]_{king} - [WR]_{man} + [WR]_{woman} \approx [WR]_{queen}$ (Mikolov et al. 2013)

Representation of Text

Representation of text is very important for performance of many real-world applications. The most common techniques are:

Local representations

- N-grams
- Bag-of-words
- 1-of-N coding

Continuous representations

Latent Semantic Analysis
Latent Dirichlet Allocation
Distributed Representations

Distributed Representation

Distributed vector representations that capture a large number of precise syntactic and semantic word relationships.

Distributed representations of words can be obtained from various neural network based language models:

Feedforward neural net language model
Recurrent neural net language model

First Proposed Model

- Four-gram neural net language model architecture (Bengio 2001)
- The training is done using stochastic gradient descent and Backpropagation
- The training complexity of the feedforward NNLM is high:
  - Propagation from projection layer to the hidden layer
  - Softmax in the output layer
- Using this model just for obtaining the word vectors is very inefficient

Improving efficiency

- The full softmax can be replaced by:
  - Hierarchical softmax (Morin and Bengio)
  - Hinge loss (Collobert and Weston)
  - Noise contrastive estimation (Mnih et al.)
  - Negative sampling (Mikolov et al)
- Mikolov et al. further removed the hidden layer: for large models, this can provide additional speedup 1000x
  - Continuous bag-of-words model
  - Continuous skip-gram model

Proposed Model

- In this work, I am proposing to use continuous skip-gram and bag of words architecture with following extension:

- I want to optimize the objective to project a common vector space to maximize correlation between the same category words

Continuous Skip-gram and Bag of words- family of log linear language model

NLP is so varied and complex, even using a extremely large corpus, we can never model all string of words. Skip-gram is a technique that allow n-gram to be stored to model the language but it allow token to be skipped.

Example

the sentence "Hi fred how was the pizza?"

becomes:

Continuos bag of words: 3-grams {"Hi fred how", "fred how was", "how was the", ...}

Skip-gram 1-skip 3-grams: {"Hi fred how", "Hi fred was", "fred how was", "fred how the", ...}

Skip-gram

Objective Function

$$\arg\max_{\theta} \prod_{w \in Text} \left[\prod_{c \in C(w)} p(c|w;\theta) \right]$$

$$\arg\max_{\theta} \prod_{(w,c) \in D} p(c|w;\theta)$$

$$p(c|w;\theta) = \frac{e^{v_c \cdot v_w}}{\sum_{c' \in C} e^{v_{c'} \cdot v_w}}$$

Hierarchical Softmax

$$p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma([n(w,j+1) = ch(n(w,j))] \cdot {v'_{n(w,j)}}^T v_{w_I})$$

Oregon State University

Some result by Milokov et al, 2013

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days x CPU cores]
			Semantic	Syntactic	Total	
NNLM	100	6B	34.2	64.5	50.8	14 x 180
CBOW	1000	6B	57.3	68.9	63.7	2 x 140
Skip-gram	1000	6B	66.1	65.1	65.6	2.5 x 125

I did not run the model on big enough dataset because of the time constraint

Some snapshot of result -name

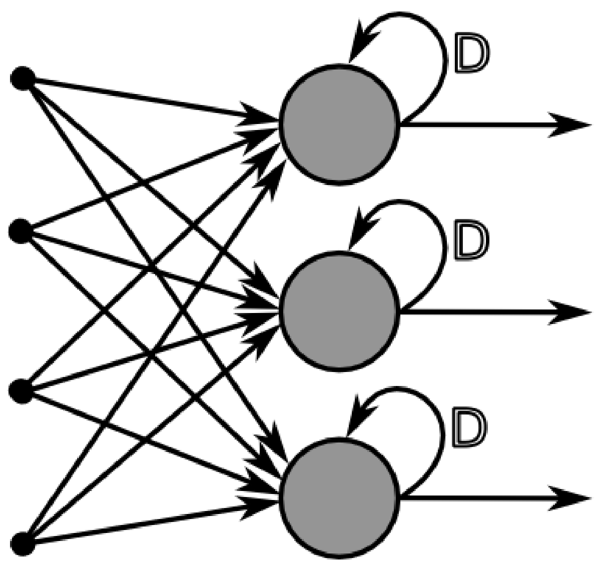
```
farhana@farhana-Inspiron-3520: ~/NER dataset/code_project
Enter word or sentence (EXIT to break): john

Word: john  Position in vocabulary: 145

                                             Word       Cosine distance
------------------------------------------------------------------------
                                            james              0.659896
                                           robert              0.646968
                                           thomas              0.633958
                                          william              0.629634
                                          richard              0.617703
                                            peter              0.609647
                                           george              0.591549
                                             paul              0.589056
                                         nicholas              0.581043
                                          anthony              0.573255
                                             hugh              0.567399
                                            henry              0.559341
                                           joseph              0.554239
                                           edward              0.553235
                                           andrew              0.549794
                                         reginald              0.549595
                                          michael              0.547040
                                          charles              0.542569
                                        archibald              0.538724
                                           martin              0.534698
                                          wilfred              0.533364
                                            nigel              0.533343
                                          stephen              0.531121
                                           arthur              0.526178
                                           dudley              0.525375
                                          patrick              0.524747
                                         alastair              0.524410
                                       evangelist              0.516333
                                           walter              0.513963
                                          knowles              0.511593
                                           samuel              0.510832
                                          kenneth              0.508504
                                          erskine              0.507395
```

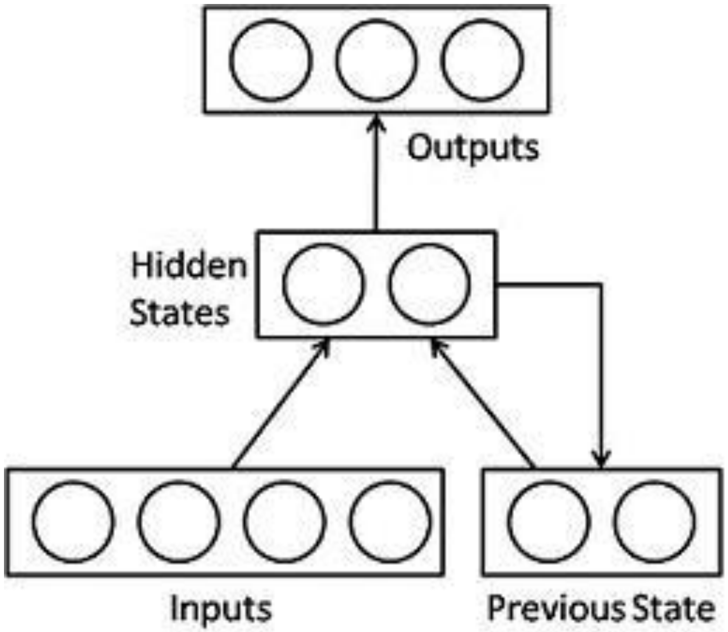
Some snapshot of result- location

```
farhana@farhana-Inspiron-3520: ~/NER dataset/code_project
Enter word or sentence (EXIT to break): washington

Word: washington  Position in vocabulary: 933

                                              Word       Cosine distance
------------------------------------------------------------------------
                                          maryland              0.488775
                                          virginia              0.468054
                                         mcclellan              0.458711
                                           lincoln              0.452632
                                      pennsylvania              0.446925
                                          yorktown              0.428010
                                      grandiflorum              0.422489
                                           potomac              0.418278
                                         roosevelt              0.411514
                                            tacoma              0.408797
                                      intelligencer             0.408757
                                            kennan              0.408533
                                      philadelphia              0.406546
                                         lafayette              0.398308
                                           peppard              0.394846
                                          illinois              0.393286
                                         jefferson              0.392677
                                              ohio              0.391940
                                            carver              0.391584
                                             meade              0.388473
                                           madison              0.387166
                                          danville              0.384288
                                         arlington              0.384238
                                          newburgh              0.381731
                                           sherman              0.381109
                                          breitman              0.380549
                                        california              0.379683
                                           peabody              0.379494
                                            pataki              0.379218
                                     massachusetts              0.378682
                                           spokane              0.377538
                                             texas              0.375799
                                       sideroxylon              0.373922
```

Recurrent Neural Network

- Considering a memory for some nodes in a neural network
- Next result will be affected by previous state. (We have directed cycle in them)





Implemented Structure

- Implemented from scratch …
- Recurrent Neural Network with these flexibilities:
  - Structural (act like):
    - Simple Neural Network
    - Elman Neural Network (a RNN)
    - Jordan Neural Network (a RNN)
    - Elman & Jordan Neural Network (a RNN)
  - Non-Linear function
    - Sigmoid Function
    - Tanh Function
  - Meta parameters & weight initialization methods
  - input & hidden Layers
  - Number of features for each words
  - etc.

Dataset

- Two different datasets
  - Informal tweets of tweeter: Detect Person, Location, Org. NE
  - Stanford dataset for NER: Detect Person NE

- Very sparse dataset with majority of zeros
  - Applying resampling with different manner and strategies

- Standard Dataset: CoNLL-2003
  - Not easily available! (some paperwork and waiting for at least 7 business days)

Learning

- Using Gradient descent with L_2 Regularization
- Using Backpropagation method
- Many Local minimums, initialization has high effect on result
  - When it goes bad, start over with another initialization
- Better to use new optimization methods: L-BFGS, AdaGrad, etc. (serious lack of time!)

Experimental Result

- NER has high accuracy since it has lots of zeros
- Resampling ones labels, reduce local minimums and preventing zero Recall
- In both dataset unpredictable & unbelievable result!!!!:

Test Accuracy	Precision	Recall	F-Measure
100%	1	1	1

- About 10% unseen data in test sets.
- It happens because of dataset
- I'm not telling that we solved NER task for sure!!!!

Future Works

- Using Standard Dataset to be able to compare results with other presented methods
- Using handy-crafted features for improving result in standard dataset
  - The result definitely will decrease in standard dataset

Thanks for your attention and time