Named Entity Recognition and Question Answering Using Word Vectors and Clustering

Zia Ahmed
zahmed@stanford.edu

Rajkiran Veluri
rveluri@stanford.edu

CS229: Machine Learning Project
Computer Science Department, Stanford University

Problem Statement

In this paper we investigate the word2Vec model as proposed by Tomas Mikilov for determining word relationships and use the word vectors to implement a Named Entity Recognition (NER) System. NER plays a key role in many Natural Processing tasks like Question Answering (QA). This is due to the fact that answers to many questions are named entities that depend on the semantic category of the expected answer. In this context, we examine the effectiveness of our NER algorithm to identify the entity category of the expected answer. In particular we study the methodology of boosting the performance of the NER system by training on pre-annotated natural language questions combined with the annotated free text data. We hypothesize that the NER system can benefit from the inclusion of the pre-labelled questions. Further we explore clustering mechanism to classify Word vectors into entity classes and discuss how clustering can be used to improve the performance of the NER system.

Introduction

The Word2Vec model, proposed by Tomas Mikolov at Google, used Skip Gram and Continuous Bag of Words (CBOW) model to create word embeddings [8]. An extension of Word2Vec is the Glove model proposed by Penninglon, which is easier to parallelize [9]. These models have enabled natural language processing tasks like NER, POS tagging to avoid manual designing of features, by using word vectors that capture the syntactic and semantic information through latent dimensional features. Still, the finer nuances of word embeddings in vectors space have not been understood fully. For Named Entity Recognition(NER), we decided to use word vectors but a comparable performance has been reported using Brown Clusters which is a hierarchical agglomerative clustering algorithm relying on maximizing the mutual information of bigrams [11]. An advantage of Brown Clusters is that it works better on rare words. Another scalable algorithm using deep learning for NER was proposed by Collobert and Weston[12]. Ratinov and Roth discuss issues in designing NER systems in [10]. In this paper we use the vector representations of words to implement a neural network based NER system, which is then utilized to aid in Question answering by reducing the answer candidates.

Word Vectors

The method used to generate word vectors is the continuous bag-of-word model (CBOW) by Mikolov et al. (2013). CBOW is a neural network model which tends to predict the target word based on the input window of context words surrounding the target word. The training process creates low-dimensional word vectors (each word is 200 dimensional) for each word in the training corpus. The word vectors which are contextually, syntactically and semantically similar tend to lie near each other in this low dimensional space, as shown in the PCA analysis of the few handpicked words from the vocabulary. Refer Figure 2 for 2D PCA representation of word vectors. We use these word representations as features to build the NER system which is described below. We implemented CBOW model and trained

using Google's dataset [1]. The algorithm performed well on smaller subsets of our data but when training on a large vocabulary (>1 M), the training time became excessively large, so we used pre-trained word vectors [1].

Named Entity Recognition

NER is a classification problem, where each input word is classified as being a Location, Person, Organization, Miscellaneous and Other (not any named entity). The algorithm uses tokenized text to train a neural network model for named entity recognition with multiple classes. The detailed annotation structure for the dataset is given at [13]. The training and the testing data for the NER algorithm is taken from CoNLL03 corpus. The data consists of sentences with one token per line and each token is associated with 5 possible labels: {O, LOC, MISC, ORG, PER} representing the classes defined above. The word vectors learned using the CBOW model were used to construct context windows that serve as input features to the neural network.

The model is implemented as single layer neural network with word embedding as the input layer feeding to feedforward algorithm. The predicted class vector is then compared to the actual class and the delta error propagates back updating the model. As the algorithm iterates through the dataset it learns both the classifier and the word representations.

The feedforward operation is given by the following set of equations

$$z = W \begin{bmatrix} x_{i-1} \\ x_i \\ x_{i+1} \end{bmatrix} + b^{(1)}$$

$$h = tanh(z)$$

$$p = g(Uh + b^{(2)})$$

Where $((x_{i-1}, x_i, x_{i+1})$ is the context window, W and U are the model parameters and g is the softmax function. The cost function to minimize is given by the following equation

$$J(\theta) = \sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(1)} log p_{\theta,k}(x^{(i)})\, A + \frac{\lambda}{2m}(\|W\|^2 + \|U\|^2)$$

Where m is the number of data samples, K is the number of entity classes and λ is the regularization parameter. The parameters are learned using stochastic gradient descent algorithm and gradient checking is used for bug-free implementation.

The evaluation of the implemented algorithm was done using the CoNLL03 *conlleval* Perl script. The script evaluates the NER system's capability of identifying named entities. It gives a clear presentation of the performance of the system on various entity categories (person, location, organization, miscellaneous and other) based on the precision, recall and F1 measures.

Results

Tuning Parameters

The parameters of the system that were tuned for higher accuracy were:

- The regularization constant (λ)
- The learning rate (α)
- The context window size (C)
- The number of iterations (epochs)

We used context window size of 3 and 5 for the model; better results were achieved with size 5. It was observed that increasing number of iterations does not necessarily increase the performance. The improvement stagnated after 40 iterations. For regularization parameter (λ =0.02) gave the best performance. The performance was decreasing for higher values of λ. The learning rate (α) was optimized at 0.075. Lower learning rate gives inferior results. The optimal values found for the tuning parameters are given in Table 1.

Parameters	Optimal Value
Epoch	40
Learning Rate (α)	0.075
Regularization (λ)	0.02
Context Window Size (C)	5

Table 1: Optimal Parameter Values for NER

	Recall	Precision	F1
LOC	85.20%	92.10%	88.52%
MISC	74.53%	91.39%	82.10%
ORG	64.58%	84.54%	73.22%
PER	76.53%	96.17%	85.23%
System	75.44%	91.73%	82.79%

Table 2: NER Evaluation Results

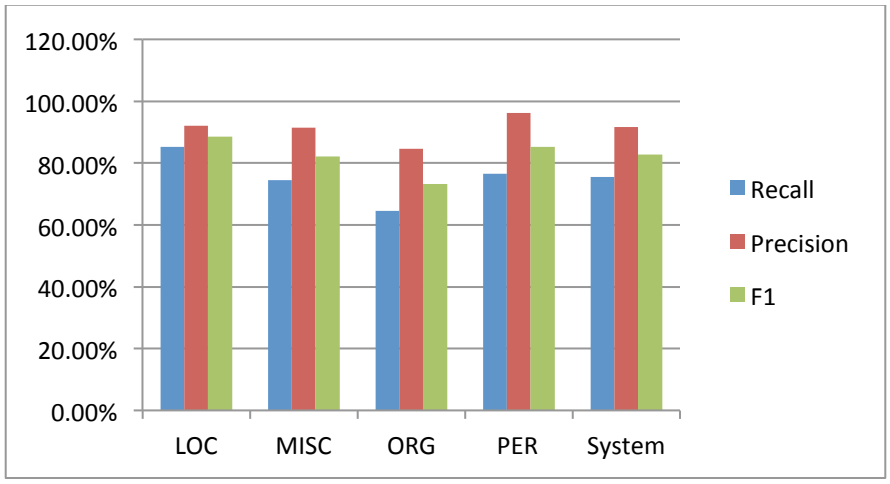

Figure 1: NER Evaluation results

Question Answering

Named Entity Recognition systems are used in a lot of NLP tasks. In particular, they play a prominent role in Question-Answering. Named Entity Recognition systems are typically used in question answering systems like AFNER to narrow down the candidate answers which match the semantic category of the selected answer. For example, the answer to the question "*Which is the Capital of France*", the system identifies the category of the expected answer to be a Location (LOC). Thus, the system will only

consider the named entities with category LOC as answers thereby affecting both the precision and performance of the overall system.

In this paper we utilize our Neural Network based NER model to identify and classify NEs in natural language questions. We hypothesize that the NER system can benefit from the inclusion of the pre-labelled questions in the training corpus. The training and testing corpus for the experiment was downloaded from [14]. The training data consists of 5500 annotated questions with categories PER, LOC and ORG. Similarly the test data consists of 500 pre-annotated questions.

The performance is baselined using the system trained on the CoNLL03 corpus and tested on the 500 test questions. The baseline results are given in the Table 3. Although the F-measure for the free text test corpus was 82.79%, the system's performance drops to 56.81% when tested on the 500 annotated questions test corpus. Next, we trained the NER system on the training corpus of 5500 pre-annotated questions and tested the resulting model both on the CoNLL03 test data and the 500 test questions. The results of the step are given in Table 4. The system performed well on the annotated questions test corpus but failed miserably for the *CoNLL03* free text test corpus. Finally, the NER system was trained on using both the CoNLL03 corpus and 5500 pre-annotated questions corpus. The performance on both the test datasets are given in Table 5. We achieved an F-measure of 83.2% on the question test data when the training data contained both the free text and pre-annotated data.

The results obtained in this work suggests that the NER system used in aiding question answering system benefits from including questions in the training corpus. To build a NER model which provides an F-measure > 80% we should build a training corpus which is a suitable mix of free text and annotated questions. As shown in Table 5, the inclusion of free text in the training data is not relevant if we have sufficient questions to train the NER system and the system is used only for Question Answering. But to build a general-purpose model the system will benefit from combination of training data.

Train			Eval	Recall (%)	Precision (%)	F-Measure (%)
Type	Sentences	Tokens	Free Text	51.37	91.73	82.79
Free text	14987	219554	Questions	51.37	63.53	56.81

Table 3: Baseline NER Results

Train			Eval	Recall (%)	Precision (%)	F-Measure (%)
Type	Sentences	Tokens	Free Text	18.18	43.17	25.58
Questions	5452	61074	Questions	72.95	89.22	80.07

Table 4: Trained on 5500 Pre-annotated questions only

Train			Eval	Recall (%)	Precision (%)	F-Measure (%)
Type	Sentences	Tokens	Free Text	76.16	90.76	82.82
Free text + Questions	20439	280628	Questions	79.03	87.84	83.20

Table 5: Trained on both free text and pre-annotated questions

Clustering word vectors for training NER

The clustering was done using K-means algorithm on the 200 dimensional word vectors generated by Word2Vec model. We constructed clusters of multiple granularities, through hierarchical clustering. The primary intuition being that clustering would give us a unsupervised automated way to increase our training data, similar to construction a NER gazetteer. We found that unigram clusters capture broad categories of entities (countries and states in a single cluster, names) and would be useful in NER if there are more number of labels. Also increasing the number of clusters gave us better separation of entities. The clusters generated were used to train the NER model. The results are given in Table 6. An important realization was that, the single word clusters do not capture context, so training n-gram clusters (setting n to be the size of the context window) could be more useful approach in clustering [5].

Training Corpus	Cluster Granularity	F-Measure (%)
Clusters	800	25.10
CoNLL03 + Clusters	800	83.40
Clusters	2000	28.34
CoNLL03 + Clusters	2000	83.56

Table 6: Testing Cluster Granularity

Further Study

Combining neural networks with word vector models for Named Entity recognition is an active field of study. Named Entity Recognition using recurrent neural networks (RNN) and Long Short Term Memory (LSTM) is also a promising future direction and better results have been achieved by it. Future directions of study for question answering would focus attention on dynamic memory networks that make the use of word vectors by combining a knowledge base (or facts) to achieve state of the art-results [6].

Acknowledgments:

We would like to thank Prof. Andrew Ng and our project mentor, Youssef Ahres, for their guidance and support during the project.

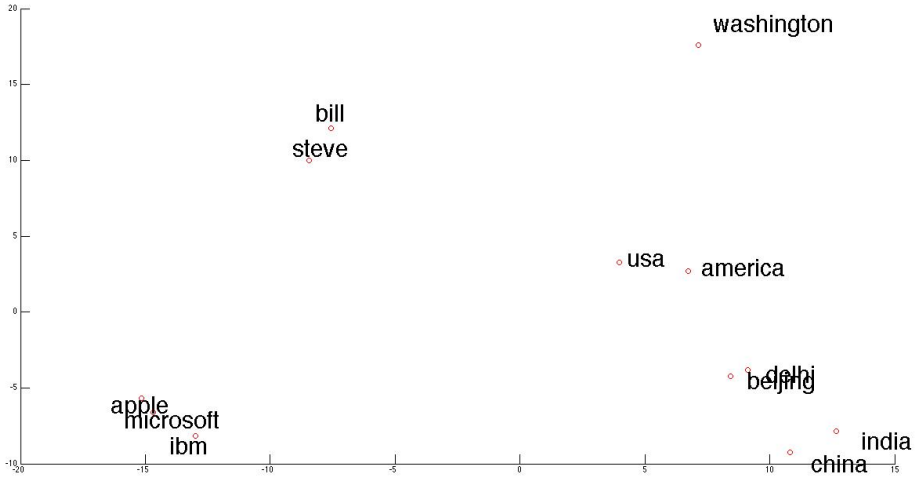

Figure 2: PCA representation of Word Vectors

References:

[1] https://code.google.com/p/word2vec/.

[2] Miller, S., Guinness, J., & Zamanian, A. (2004, May). Name Tagging with Word Clusters and Discriminative Training. In *HLT-NAACL* (Vol. 4, pp. 337-342).

[3] Siencnik, S. K. (2015, May). Adapting word2vec to Named Entity Recognition. In *Nordic Conference of Computational Linguistics NODALIDA 2015* (p. 239).

[4] Mendes, A. C., Coheur, L., & Lobo, P. V. (2010, May). Named Entity Recognition in Questions: Towards a Golden Collection. In *LREC*.

[5] Lin, D., & Wu, X. (2009, August). Phrase clustering for discriminative learning. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2* (pp. 1030-1038). Association for Computational Linguistics.

[6] Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., ... & Socher, R. (2015). Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

[7] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

[8] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[9] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, *12*, 1532-1543.

[10] Ratinov, L., & Roth, D. (2009, June). Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning* (pp. 147-155). Association for Computational Linguistics.

[11] Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., & Lai, J. C. (1992). Class-based n-gram models of natural language. *Computational linguistics*, *18*(4), 467-479.

[12] Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160-167). ACM.

Data

[13] http://www.cnts.ua.ac.be/conll2003/ner/annotation.txt
[14] https://qa.l2f.inesc-id.pt/wiki/index.php/Resources
[15] http://mattmahoney.net/dc/text8.zip