Add Named Entity Recognizer to Convolutional Neural Network

Guangyu Lin
UTEID: gl8429
glin@utexas.edu

Ge Gao
UTEID: gg24984
gegao1118@utexas.edu

Huihuang Zheng
UTEID: hz4674
huihuang@utexas.edu

Abstract

In this work, our assumption is that adding Named Entity Recognizor (NER) [12] as a feature for Convoluional Neural Network can improve accuracy for NLP tasks involving named entities. To verify our assumption, we try to solve TREC question classification task [8]. Our method adds Named Entity Recognizor (NER) as one dimension number to word2vec expression, then the modified expression is used for training a CNN. Our experiment illustrated that our CNN + NER model can improve the accuracy of question type classification task compared to only CNN method. Although our method still can not get comparable performance of the state-of-the-art work, Our method is easier to train because the state-of-the-art work needs a large number of hand-code resources.

1. Introduction

Automatic question answering is an important information retrieval task in natural language processing and it is more challenging than purely searching task because it needs to "understand" what is asked before searching for answers. And to make machines "understand" a question is as challenging as it sounds like. One method that could help solve this problem is called Question Classification(QC). Question Classification(QC) is a task that given a question, some classifier will map this question to one of k limited classes, so that some semantic constraint will be put on this question, which provides machine with some basic information about the question which could hopefully help machine "understand" questions.

In 2000's TREC competition [8], participants were requested to develop a system to classify English questions based on some question categories. And after that, many remarkable results have been achieved. One of them is Li and Roth's work. Li and Roth develops a hierarchical classifier based on SNoW learning architecture(Carlson et al, 1999 [4]; Roth, 1998 [13]) to solve the question classification task.

With the development of deep learning, machine learning models have been applied to NLP tasks and it turns out that many of these models significantly improved performance of original models without deep learning methods. One example is Yoon Kim's [7] work that applies convolutional neural networks (CNN) to sentence classification.

Since named entity is an important part of questions, especially when number of question types is large, but previous work did not make much use of the semantic information of a named entity. For example the named entity might be either a location or a person. This work makes use of Stanford's Named Entity Recognizer (NER) [12] and adds extra labels to vectors generated by *word2vec*. Then we train convolutional neural network model with these new vectors. Our goal is to improve the performance of the CNN model for question classification, especially for questions related to named entity semantic meanings.

The main contribution of this work is that our method, which adds Name Entity Recognizer (NER) as one extra dimension for each *word2vec*'s output vector, can improve top-1 accuracy in a 50-classes question classification task – TREC fine types, which demonstrates that by combining NER and CNN, our method can improve CNN classification accuracy. Although our method still cannot outperform the state-of-the-art method which doesn't apply neural networks, our CNN classifier is easier to deploy because the non-neural approach method uses a large number of hand-coded resources.

2. Problem Definition and Algorithm

2.1. Task Definition

TODO

2.2. Algorithm

2.2.1 Named Entity Recognition

Named Entity Recognition (NER) [5]labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. All named entities are self-explanatory in Tabel 1. As named entity recognition is frequently a prelude to identifying relations in Information Extraction, it can also contribute to

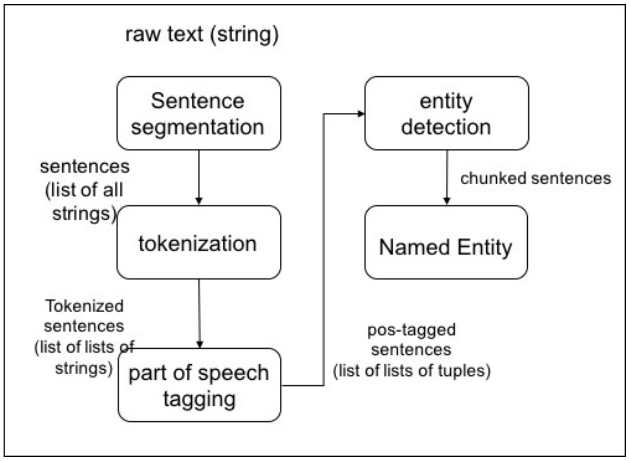

Figure 1. Simple Pipeline Architecture for an Named Entity Recognition System.

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408

Figure 2. Simple Pipeline Architecture for an Named Entity Recognition System.

other tasks. For example in our Question Answering(QA), we can improve the accuracy of our question classifiers by recovering not whole pages, but just those parts which contain an answer to the user's question. For example, we have a question,

"How did serfdom develop in and then leave Russia ?"

Although this is a particular question about a description of manner, this is also a question to query something related to Russia, which is a named entity - LOCATION. While searching the correct answer, it will help us skip very wrong answer if we can combine with the location named entity. Therefore we would like to use Named Entity Recogniser to preprocess our data in order to help our classifier skip very wrong answer.

NE Type	Examples
ORGANIZATION	Georgia-Pacific Corp., WHO
PERSON	Eddy Bonte, President Obama
LOCATION	Murray River, Mount Everest
DATE	June, 2008-06-29
TIME	two fifty a m, 1:30 p.m.
MONEY	175 million Canadian Dollars, GBP 10.40
PERCENT	twenty pct, 18.75 %

Table 1. NER

In Figure. 1, we present a simple pipeline architecture for an named entity recognition system. However, we would like to use Stanford Named Entity Recognizer (NER) to retrieve named entity from our sentences. Because they train a large corpus and have a good performance.

2.2.2 Word2Vec

Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. We use word2vec to turn text into a numerical form that convonlutional neural network can understand. The output of the word2vec is a vocabulary in which each item has a vector attached to it. We can also easily store this model and reload this model in the later. In order to visualise the result of word2vec, we present an example on the website. Here?s a list of words associated with "Sweden" using Word2vec, in order of proximity, which shows in Figure. 2

The nations of Scandinavia and several wealthy, northern European, Germanic countries are among the top nine. [1] In our experiment, we would like to use word2vec to train our sentence and get each word as a vector v, then prepare it for convolutional neural network.

2.2.3 Convolutional neural network

After we get vector v for each word where the $v \in \mathbb{R}^k$ is the k-dimensional word vector, we use Kim's CNN architecture. A sentence of length n (padded where necessary) is represented as concatenation of word vectors:

$$v_{1:n} = v_1 \oplus v_2 \oplus v_3 ... \oplus v_n$$

where v_i is corresponding to the i-th word in the sentence.

In general, let $v_{i:i+j}$ refers to the concatenation of word vectors $v_i, v_{i+1}, ..., v_{i+j}$. We will describe how one filter extracts feature and then discuss the CNN with multiple convolutional filters. A convolutional filter of Convolutional Neural Network (CNN) [7] takes fixed size of input of $\mathbb{R}^{hk}$. A convolutional filter $w \in \mathbb{R}^{hk}$ is applied to each possible window of words of the sentence

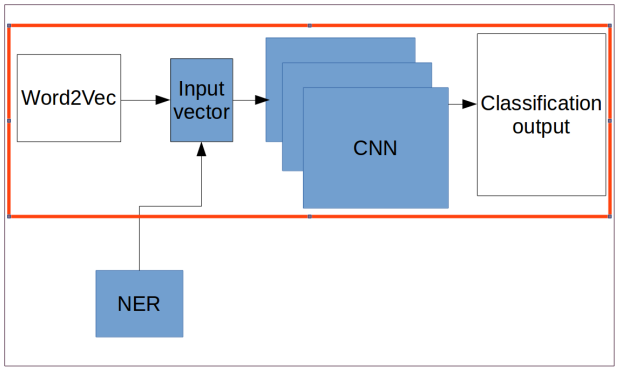

Figure 4. Our Approach: Red Rectangle marked the traditional method and Blue Boxes are where we changed

$\{x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}\}$. The features are extracted as

$$c_i = f(wx_{i:i+h-1} + b)$$

So convolutional layer produce a feature map:

$$c = [c_1, c_2, ...c_{n-h+1}]$$

Then, we applied a max pooling over the feature map and take the maximum value:

$$\hat{c} = \max\{c\}$$

The idea is to capture the most important feature.

Above we describe the process how one feature was extracted by convolutional neural network by one filter. In fact, the model has multiple filters so it extracts multiple features. These features from the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over labels. Figure 3 shows an

2.2.4 Our Approach

We add *Name Entity Recognizor* (NER) to the word expression v. Let our word vector be $v' \in \mathbb{R}^{k+1}$. We are going to add one dimension to *word2vec* expression. Let

$$v' = [v, e]$$

where v is the output vector of *word2vec* and e is the label number indicating the named entity. Then we will modify CNN architectures: add one input dimension and keep others the same as previous CNN. Suppose previous CNN convolutional filters take input of $\mathbb{R}^{kh}$ and we will change input size into $\mathbb{R}^{(k+1)h}$.

Figure 4 shows an figure illustration of our approach.

3. Experiment

3.1. Methodology

3.1.1 Dataset

Our method was evaluated by comparing accuracy in a question classification task named TREC [9] with previous researchers' results. The task involves classifying a question into k question types. They defined a two-layered taxonomy, which represents a natural semantic classification for typical answers in the TREC task. The hierarchy contains 6 coarse classes (ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION and NUMERIC VALUE) and 50 fine classes. Each coarse class contains a non-overlapping set of fine classes. We choose this dataset because of two reasons. The fact that Named Entity Recognizer can classify person, location may help us in this task. Second, in Kim's paper [7] and Li's paper [9], they also used this database and reported previous accuracy so it will be easy for us to compare. The TREC dataset has 5500 annotated questions for training and 500 questions as the test set.

3.1.2 Setup

We used *word2vec* trained model from Mikolov [10] on 100 billion words of Google News and from [11]. We built our framework using NER from Stanford NER software [5]

and deep learning tool Theano [2]. Since Kim's work[7] has Github published code using Theano (`https://github.com/yoonkim/CNN_sentence`), our method which combines CNN and NER was extended from their code.

We compared our method with Kim's method of using CNN and previous best methods without using CNN in the TREC dataset. By comparing CNN and our CNN + NER method, we can evaluate the contribution of NER features in this task. In addition, because Kim's method didn't outperform previous methods without CNN, we need to compare previous methods.

3.1.3 Evaluation

In this paper, we use same metric as previous researchers. We count the number of correctly classified questions by two standards. Top-1 accuracy and top-5 accuracy. We define

$$I_{i,j} = \begin{cases} 1 & \text{Correct label of the } i\text{-th question is output within rank } j \\ 0 & \text{otherwise} \end{cases}$$

Here, classifiers will give j labels for the i-th question. If the correct label for question i is among those j labels, $I_{i,j}$

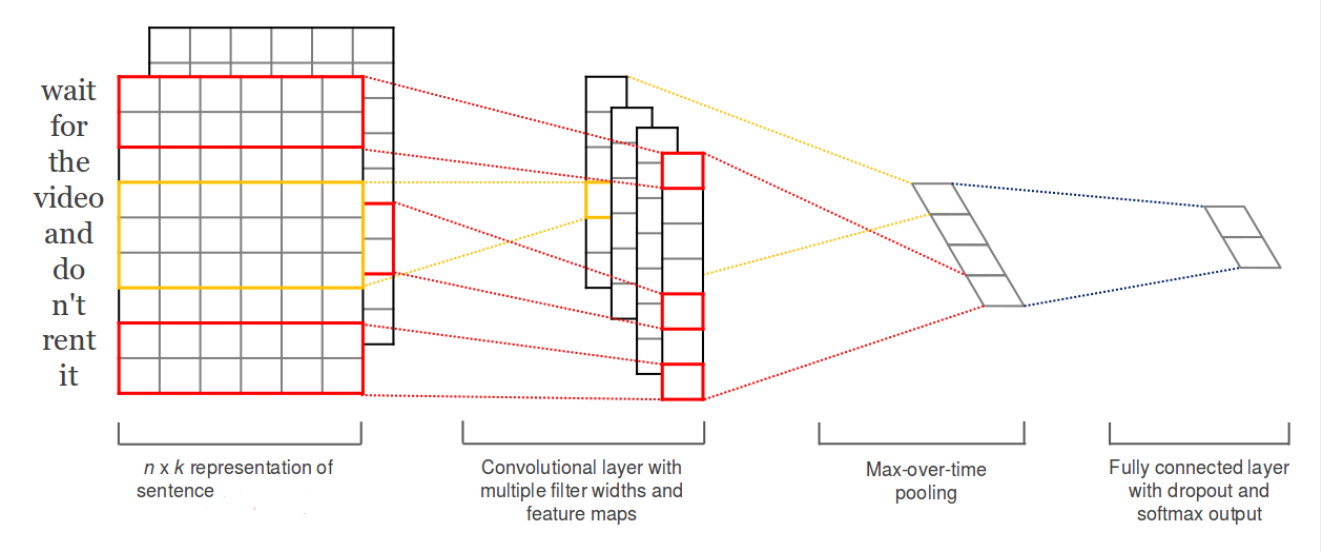

Figure 3. CNN model example

Method	Top-1 Accuracy	Top-5 Accuracy
Hierarchical classifier[8]	91.0	98.8
SVM_S [15]	95.0	-
CNN [7]	91.8	100.0
CNN + NER (ours)	91.8	100.0

Table 2. Top-1 and top-5 accuracy for TREC coarse question types.

Method	Top-1 accuracy	Top-5 accuracy
Hierarchical classifier[8]	84.20	95.00
SVM_S [15]	90.8	-
CNN [7]	80.2	90.60
CNN + NER (ours)	81.0	90.60

Table 3. Top-1 and top-5 accuracy for TREC coarse question types.

will be set to 1. Top 1 accuracy is defined as $j = 1$, which is a usual precision definition. Top 5 accuracy is defined as $j = 5$, which allows multi-labels for classifiers. Formally, if we have n test samples, top-k accuracy is defined as

$$P_k = \frac{1}{n}\sum_{i=1}^{n} I_{i,k}$$

3.2. Experimental Results

We reimplemented Kim's CNN method as baseline. We train Kim's CNN and our CNN + NER in 2000 epochs. Then, we record the best accuracy in the 2000 epochs. Our result uses non-static CNN in Kim's paper because it got best result than other Kim's CNN model in this task. We compared with two methods without using CNN: 1, Hierarchical classifier by Li and Roth[8] uses features unigram, POS, head chunks, NE, and semantic relations. 2, SVM_s uses ni-bi-trigrams, wh word, head word, POS, parser, hypernyms, and 60 hand-coded rules as features from Silva et al [15], which is the state-of-the-art in the best of our knowledge. Our test result for coarse classes was shown in table 2. Our test result for fine classes was shown in table 3

3.3. Discussion

In table 2, we can see our CNN + NER model doesn't improve accuracy from Kim's only CNN model. However, when it is in a more complexed situation, the fine task in table 3, we can see our model can slightly improved the top-1 accuracy (81.0 vs 80.2 in only CNN model).

Comparing to non-neural approaches methods, which needs manually engineered features. CNN's approach is easier to train. Our method has higher top-1 and top-5 accuracy than the Hierarchical classifier method in coarse task. However in fine task, Li and Roth's method outperforms ours. According to the fact that we add NER's feature to CNN and improved the accuracy of fine task, we think that demonstrates the human defined features are helpful in fine types classification.

Our CNN + NER model has not got comparable accuracy to the work of Silva et al. [15]. However, their classifier involves a large number of hand-code resources. For example, they have 60 hand-coded rules. CNN's training is simpler because we don't need those manually engineered features.

4. Related Work

4.1. Question classifiers

In the "Learning Question Classifier" paper (Li and Roth, 2002 [8]), in terms of the question classification task, the number of question types(k) could be either six or fifty depending on different classification criteria.

Li and Roth developed a machine learning method to classify questions which is guided by a layered semantic hierarchy of answer types. They made use of a sequence of two simple classifiers to do the classification. The first classifies questions into coarse classes (six in total) and the second classifies questions into fine classes (fifty in total), which is dependent on the first one. Here is the structure of the hierarchical classifier by Li and Roth.

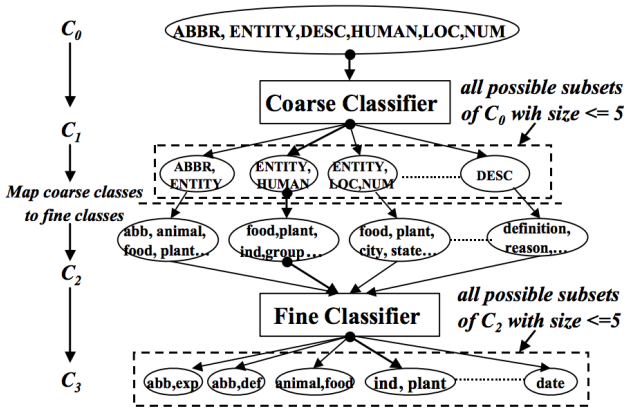

Figure1: The hierarchical classifier

Figure 1 shows the basic classification process by classifier developed by Li and Roth. A question will always be processed along a top-down path to be classified. And after the classification process, question type label(s) will be attached to the question.

In Li and Roth's classifier, each question is analyzed as a list of features so that they could be trained and tested for learning. They extracted some primitive features like words, pos tags and chunks (Abney, 1991 [3]), named entity as well as some semantically related words. Based on these features, they compose and make some more complex features. Also, they make a semantically related word list for each of most question types. For example, "far" is in the semantically related words of "distance" so that if there is an occurrence of "far", and then the sensor for "distance" will be activated and the feature will be extracted.

A point that is worth pointing out is that there might be ambiguity for some specific questions. For example, a question like "What do bats eat" could either be classified to belong to food type or animal type, and both them make much sense. To solve this, Li and Roth allow multiple type labels to be attached to a single question.

4.2. CNN on sentences classification

Convolutional neural networks (CNN) model is a deep learning method which has achieved remarkable results in many fields. A CNN model was developed by Yoon Kim [7] for sentence classification which accepts word vectors as input and generates type labels as output.

Convolutional neural networks was originally invented for tasks in computer vision fields and was proved to be also effective in many NLP tasks. In Kim' model, they train one layer of filter on top of word vectors generated by word2vec developed by Google (Mikolov et al 2013 [10]). In Kim's model, there are "static" and "nonstatic" models for model variation, where "static" means that the vectors are directly from word2vec and for "nonstatic", those vectors will also be tuned for each data set.

5. Future Work

This time we tried add NER features to CNN and improved CNN question classifier slightly, we think if we add more features, the CNN classifier can be more accurate.

6. Conclusion

In this work, we add Named Entity Recognizor (NER) as one dimension number to *word2vec* expression. The modified expression can be used for training a CNN for NLP tasks involving named entity. Our experiment shows that our CNN + NER model can improve the accuracy of question type classification task compared to only CNN method. Although our method still can not get comparable performance of the state-of-the-art work, the state-of-the-art work needs a large number of hand-code resources. Our method is easier to train.

References

[1] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah. Comparative study of caffe, neon, theano, and torch for deep learning. *arXiv preprint arXiv:1511.06435*, 2015.

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[3] R. C. Berwick, S. P. Abney, and C. Tenny. *Principle-based parsing: Computation and psycholinguistics*, volume 44. Springer Science & Business Media, 1991.

[4] A. Carlson, C. Cumby, J. Rosen, and D. Roth. The snow learning architecture. Technical report, Technical report UIUCDCS, 1999.

[5] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[6] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[7] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[8] X. Li and D. Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational*

linguistics-Volume 1, pages 1–7. Association for Computational Linguistics, 2002.

[9] X. Li and D. Roth. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249, 2006.

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

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. word2vec, 2014.

[12] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[13] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *AAAI/IAAI*, pages 806–813, 1998.

[14] D. Roth, G. K. Kao, X. Li, R. Nagarajan, V. Punyakanok, N. Rizzolo, W.-t. Yih, C. O. Alm, and L. G. Moran. Learning components for a question-answering system. In *TREC*, 2001.

[15] J. Silva, L. Coheur, A. C. Mendes, and A. Wichert. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154, 2011.

[16] E. M. Voorhees and D. Tice. Overview of the trec-9 question answering track. In *TREC*, 2000.