2018 12th IEEE International Conference on Semantic Computing

# Relation Extraction using Multi-Encoder LSTM Network on a Distant Supervised Dataset

Siddhartha Banerjee
Yahoo!
Sunnyvale, CA, USA
siddb@yahoo-inc.com

Kostas Tsioutsiouliklis
Yahoo!
Sunnyvale, CA, USA
kostas@yahoo-inc.com

***Abstract*—Relation extraction techniques are used to find potential relational facts in textual content. Relation Extraction systems require huge amount of training data to model semantics of sentences and identify relations. Distant supervision, often used to construct training data for relation extraction, produces noisy alignments that can hurt the performance of relation extraction systems. To this end, we propose a simple, yet effective, technique to automatically compute confidence levels of alignments. We compute the confidence values of automatically labeled content using co-occurrence statistics of relations and dependency patterns of aligned sentences. Thereafter, we propose a novel multi-encoder bidirectional Long Short Term Memory (LSTM) model to identify relations in a given sentence. We use different features (words, part-of-speech (POS) tags and dependency paths) in each encoder and concatenate the hidden states of all the encoders to predict the relations. Our experiments show that a multi-encoder network can handle different features together to predict relations more accurately (~9% improvement over a single encoder model). We also conduct visualization experiments to show that our model learns intermediate representations effectively to identify relations in sentences.**

## I. Introduction

Relation extraction, a text classification task, is used to identify the relation between a pair of entities mentioned in a span of text. Relation extraction has been found to be very effective in expanding knowledge-bases automatically as well as in Question Answering (QA). Manually generating training examples (sentences and corresponding relation labels) for the ever-increasing number of relations in Knowledge-Bases (KB's) such as Wikidata or Freebase is time-consuming and expensive. As a result, distant supervision has emerged as a popular technique to automatically construct training data for relation extraction learning tasks. The underlying assumption of all distant supervision techniques is the following: If two entities are connected by a relation, a sentence that contains both the entities must describe the same relation. However, such techniques suffer from the noisy data problem as the aligned pairs of relation triples from KB's and sentences often contain erroneous alignments.

Recently, several neural network-based techniques have been proposed to address the problem of relation extraction. Neural network-based techniques are capable of automatically learning representations of sentences containing entity pairs and predicting relations in the text. Most of the techniques [1], [2] have been developed using Convolutional Neural Networks (CNN) [3] as the underlying learning structures. However, a recent analysis [4] shows that Recurrent Neural Networks (RNN) generally perform better than CNN's on relation extraction tasks. To the best of our knowledge, all the above mentioned techniques consider only the word sequences in the sentences as inputs to the network. However, a number of other features can potentially be used to augment word features such as information from Named-Entity Recognition (NER), dependency paths and Part-Of-Speech (POS) tags.

In this work, we address the above mentioned concerns. First, we build a distant supervised dataset for relation extraction by mapping triples from DBPedia [5] and sentences from Wikipedia. To reduce the effect of noisy labels, we compute confidence values for the samples based on co-occurrence statistics of dependency paths in the sentences and relations. We use the confidence values as sample weights during model training allowing the models to automatically adjust parameters depending on importances of individual data instances. Second, we propose **MEM** (**M**ulti-**E**ncoder **M**odel) that uses three simultaneous Long-Short Term Memory (LSTM)[1] [6] units to encode information using features from words, POS tags and dependency paths of the sentences. The hidden states from each of the encoders are concatenated and used to predict the relation using softmax activation [7].

We conduct several experiments to evaluate the effectiveness of our proposed techniques. First, we compute the accuracy of MEM on test instances and compare it to a simple LSTM model that encodes information only from sequences of words. Our experiments show that MEM achieves ~9% improvement over a single LSTM encoder model in terms of accuracy. Second, we visualize final hidden layer representations using T-SNE [8] to show that the model is able to differentiate between different relations appropriately.

## II. Proposed Approach

In this section, first, we describe our approach to automatically construct training data using distant supervision followed by computing confidence values of each sample using co-occurrence statistics. Next, we provide details on our MEM model used to classify relations.

### A. Automatic Training Data Construction

Distant supervision has been successfully used for many relation extraction tasks. In DeepQA [9], researchers used distant supervision successfully by aligning relations from DBPedia [5] with sentences from Wikipedia articles. DBPedia

[1]LSTM is a specific type of a RNN cell.

0-7695-6360-0/18/$31.00 ©2018 IEEE
DOI 10.1109/ICSC.2018.00040

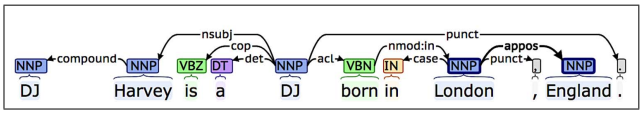



Fig. 1: Dependency graph obtained using Stanford CoreNLP

(or any other Knowledge Base (KB)) consists of triples mentioning relations between pairs of entities. Any triple can be represented as $(e1, e2, rel)$ where $e1$ and $e2$ are two entities related by a relation $rel$. For example, the following DBPedia triple: (*Square Shells*, *Kurt Vile*, *artist*) represents the information that the entities *Square Shells* and *Kurt Vile* are related by an *artist* relationship. DBPedia triples are retrieved from Wikipedia infoboxes[2]. Naturally, there is a high chance of the relation between the entities being explicitly stated in the content of the Wikipedia article. For example, the first sentence of the Wikipedia article on *Square Shells* is as follows:

> *Square Shells* is a limited edition EP by American indie rock musician *Kurt Vile*, released on May 24, 2010 on Matador Records.

The sentence contains both the entities in the triple. Therefore, it can be aligned to the specific triple from DBPedia. We create a distant supervised dataset by aligning sentences and corresponding relation triples. Following DeepQA [9], we align the first sentence in the Wikipedia article that contains both the entities[3]. However, sentences containing the two entities might not always represent the correct relation as described in the KB. For example, consider another triple *(Dalpat Singh Paraste, Shahdol, birthPlace)*. The first sentence in Wikipedia that contains both the entities is the following:

> *Dalpat Singh Paraste* (30 May 1950 – 1 June 2016) is four times MP from *Shahdol*.

Clearly, the above sentence and triple pair should not be aligned. Manually removing such erroneous labels is time-consuming and costly. Therefore, it is necessary to automatically judge confidence levels of the alignments.

**Co-occurrence based confidence computation:** Consider the following sentence:

> *DJ Harvey* is a DJ born in *London*, England.

aligned to the triple: *(DJ Harvey, London, birthPlace)*. The dependency parse for the above mentioned sentence is shown in Figure 1. We use three information elements from the dependency graph:

1) Root node [In the example, the word *DJ*]
2) Path to $e1$ [<l-nsubj>]
3) Path to $e2$ [<r-acl>$\rightarrow$ born $\rightarrow$ <r-nmod:in>]

We include indicators $l$ and $r$ (left and right) in the paths to differentiate between path elements to $e1$ and $e2$. We formulate a strategy using dependency patterns to generate confidence scores for our alignments. Let us consider the following case where an entity $e1$ was both *born* and had *died* in the same location $e2$. The sentence: $e1$ was born in $e2$ will be aligned to both relations *birthPlace* and *deathPlace*, with the second alignment being erroneous. Our confidence computation should assign a high value to the first alignment and a very low value to the second.

Let us represent the set of combined information elements as $dep_{elem}$ and $freq(dep_{elem})$ as the total frequency of the dependency element. In the example above, the $dep_{elem}$ can be represented as $(< l-subj >, DJ, < r-acl > \rightarrow born \rightarrow < r-nmod:in >)$ by concatenating all the components of the dependency graph we are interested in. We want to compute the probability of a relation $rel$ given a specific $dep_{elem}$.

$$p(rel|dep_{elem}) = \frac{freq(rel, dep_{elem})}{freq(dep_{elem})} \quad (1)$$

As shown in Equation 1, $freq(rel, dep_{elem})$ refers to the co-occurrence frequency of relation $rel$ and $dep_{elem}$. In other words, it refers to the number of sentences where the dependency $dep_{elem}$ exists between two entities related via relation $rel$. We can compute the confidence distribution of the relations given $dep_{elem}$ as shown in Table I. As can be seen, it is highly likely that with the specific $dep_{elem}$ above, the *birthPlace* relation and their corresponding alignments (that includes the verb *born*) are the most reliable, while the other relations have minimal confidence scores. We store the values of $freq(rel, dep_{elem})$ and $p(rel|dep_{elem})$ along with each alignment.

| Relation | Frequency | Confidence |
|---|---|---|
| deathPlace | 3 | 0.05 |
| residence | 2 | 0.04 |
| birthPlace | 61 | **0.85** |
| nationality | 1 | 0.02 |
| hometown | 2 | 0.04 |

TABLE I: Confidence values of different relations for a specific $dep_{elem}$.

### *B. MEM: Multi-Encoder Model*

Our proposed neural network architecture, **MEM**, to predict relations from text is shown in Figure 2. As can be seen from the figure, MEM uses three encoders to encode information from three feature sequences. An RNN with Long-Short Term Memory units (LSTMs) [6] is applied to consecutively process the sequential inputs. More specifically, we used the bidirectional LSTM [10] cell as it performed better than the regular LSTM cell. We encode the following sequences:

1) **Dependency features:** The dependency element obtained earlier using the parse is fed as a sequence to the encoder. We create the sequence by concatenating the path to $e1$, the root word, and the path to $e2$ in order as shown earlier.
2) **Word features:** We include sequence of words between the entities in the sentence as inputs for processing in the second LSTM encoder. We use an embedding layer over this to convert the raw word indices to distributed representations.
3) **POS features:** We include sequence of POS tags of the words between the entities in the sentence as inputs for processing in the third LSTM encoder.

[2]https://en.wikipedia.org/wiki/Help:Infobox

[3]We also use Wikipedia internal links to augment information on entities and increase coverage of alignments.

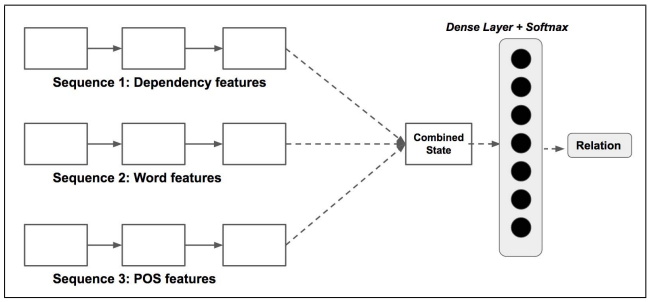



Fig. 2: Multi-Encoder Model to predict relations from text.

The final cell states of the LSTM encoders are combined (using concatenation) to create a combined cell state. Therefore, if $c_1$, $c_2$ and $c_3$ are the three cell states from the first, second and third encoder respectively, the combined state $C$ is obtained by concatenating the three representations together in the following manner:

$$C = [c_1, c_2, c_3] \tag{2}$$

We restrict each sequence to a maximum of 10 elements. Sequence elements that are closer to the entities in the sentence have been found to be more important for relation determination [1]. Therefore, for longer sequences (>10 elements), the first and the last 5 elements were used to train the model. Softmax activation function [7] was used in the final dense layer to predict the relations. We also use the confidence values computed during the training data generation phase by feeding them as sample weights when training the model using a categorical cross-entropy loss function [11]. Furthermore, due to the imbalanced nature of our dataset, we apply balanced class weighting to avoid bias when training the model.

In addition to the regular MEM model, we also explored a modified model with attention (we will refer to the model as **MEM-ATT**) following a soft-attention mechanism [12]. Both our models are fairly flexible as a number of different sequential features can be combined together to produce a comprehensive representational state that can learn to predict relations effectively.

## III. Experimental Results

In this section, we describe our dataset characteristics followed by the results of our experiments.

### A. Dataset characteristics:

We align triples from DBPedia[4] and Wikipedia [5] to obtain a total of 758,662 alignments. We normalized all entities retaining entity types (PERSON, ORGANIZATION, etc.)[6] and proper nouns (as NNP tokens) in the dataset. The alignments resulted in a total of 483 unique relations. In this paper, we only consider the 100 most frequent relations. All the remaining relations are grouped into a separate "NA" class. The top 100 relations contribute 95.9% records of the dataset. We randomly split the dataset into 70, 20, 10 percent to create the training, validation and test sets respectively.

| Model | Accuracy@1 | Accuracy@5 |
|---|---|---|
| MEM | **72.62** | 94.27 |
| MEM-ATT | 72.44 | 94.27 |
| MEM-BASIC | 71.11 | 91.48 |
| S-LSTM | 63.89 | 91.26 |

TABLE II: Performance comparison of various models on the relation classification task.

### B. Comparison between models

**General settings:** For the embedding layers, we use 300 dimensional Glove embeddings [13] to initialize the word vectors. We used a vocabulary size of 30000 words[7]. We used the Keras framework[8] to implement the neural networks.

**Models:** We compare the performance of our proposed techniques with the following two baseline models:

1) **S-LSTM**: Single bidirectional LSTM encoder that encodes only sequences of words between the entities.
2) **MEM-BASIC**: Multi-encoder model similar to MEM, without using confidence weights.

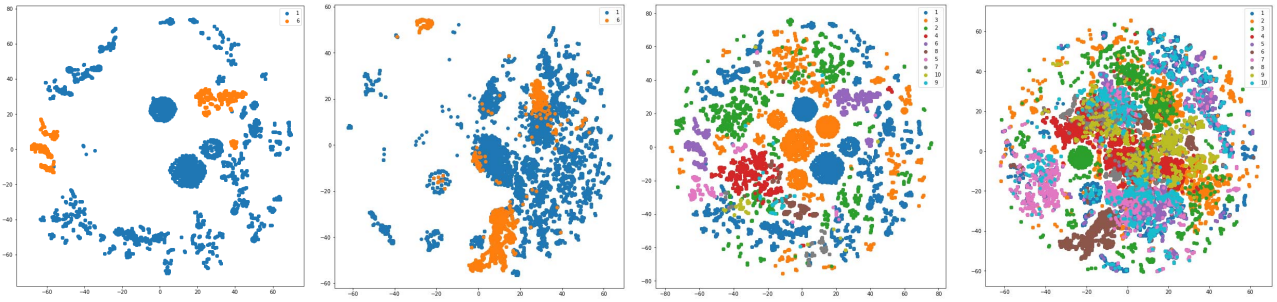
The results of relation classification on the test instances are shown in Table II. We show the accuracy at two levels (1 and 5). Accuracy@1 implies the percentage of test instances where the model predicts the relation correctly. Accuracy@5 implies the percentage of test instances where the model predicts the correct relation among the top 5 most probable classes. As can be seen from the table, MEM performs better than the other models. MEM-ATT (with attention) performs slightly worse than MEM. However, accuracy@5 is the same for both MEM and MEM-ATT implying that both the models can figure out the correct relation among their top 5 predictions in ~94% of the cases. Both MEM and MEM-ATT perform significantly better than S-LSTM (~9% improvement in accuracy@1) showing that the multi-encoder network is able to capture the semantics from multiple types of features more effectively than a single bidirectional encoder model. Furthermore, differences with MEM-BASIC show that using confidence values boosts up the accuracy as the parameter training is more effective with sample weights.

**Heldout dataset**: We randomly sampled 1000 examples (with 82 unique relations) from our test set and obtained editorial judgments. 881, 53 and 66 were marked as *yes*, *no* and *maybe*, respectively. We obtain an accuracy of **91.5**% using the *MEM* model on the 881 (yes) instances showing that the model has a very high accuracy on the manually tagged true examples. The MEM-BASIC model produces an accuracy of 89.4% (~2.3% reduction compared to MEM) on this dataset indicating that using sample weights is beneficial for parameter learning for this model.

### C. Visualizing hidden representations

The ability of a model can be realized from the representations learned by the final hidden layer of the network.

[4]DBPedia version dated 2016-10
[5]Wikipedia dump as of 20 April, 2017.
[6]Entity types detected by the Stanford NER tagger.
[7]This includes the tokens from the dependency path and POS tags.
[8]https://keras.io/



(a) 2 relations with thresholds. (b) 2 relations without thresholds. (c) 10 relations with thresholds. (d) 10 relations without thresholds.

Fig. 3: T-SNE scatter plots. In (a) and (b), instances of relations *team* (orange) and *birthPlace* (blue) are shown. In (c) and (d), the 10 most frequent relations [*birthPlace, country, isPartOf, location, deathPlace, team, nationality, city, state, hometown*] in the test set are shown.

Therefore, we create a truncated version of MEM by removing the last activation (softmax) layer and load the model with the trained weights. Thereafter, we predict the hidden layer representations[9] for the test instances with the truncated model.

We use Principal Component Analysis (PCA) [14] to reduce the dense layer representation to 50 dimensions and then use T-SNE to further reduce it to 2 dimensions to visualize the final representations as shown in Figure 3. In Figures 3a and 3b, we show the T-SNE output of the instances labeled with their actual relation labels (*birthPlace* (blue) and *team* (orange)). Figure 3b shows the T-SNE output for all the test instances belonging to both the above mentioned classes. By contrast, 3a shows the features corresponding to test instances that are more confident. We filter test instances using $p(rel|dep_{elem}) >= 0.5$ and $freq(rel, dep_{elem}) >= 20$. As can be clearly seen, the T-SNE features are well separated in 3a compared to 3b. Our proposed model MEM has been able to learn representations accurately to separate the two relations. Furthermore, the model performs much better on instances having higher confidence values implying that sample weighing is effective. We also show the T-SNE output with the test instances belonging to the top 10 most frequent sections. Figure 3c shows the output with the more confident instances. As can be clearly seen, the predicted representations (reduced to $2D$-space) of the final layer are better separated compared to Figure 3d (contains all instances without thresholds).

In the future, we plan to explore further on this dataset using human editorial judgments on the generated instances and compare with other CNN-based techniques that are currently used for relation extraction.

## IV. Conclusions and Future Work

In this paper, we presented a multi-encoder bidirectional LSTM-based model, MEM, that can encode information from multiple feature sequences to predict relations. We also described a simple co-occurrence based strategy to compute confidence scores for training instances obtained using distant supervision. We use the confidence scores as sample weights to train the model. We conduct experiments to show that our multi-encoder model is more effective than a single-encoder model in terms of accurately classifying relations in the text. Furthermore, we also perform visualization experiments using T-SNE and show that the representations learned by our model can differentiate instances belonging to different relations. In the future, we plan to combine CNN and LSTM networks to investigate new models for relation extraction.

[9]The representations are 1800 dimensional – 300 (embeddings) * 2 (bidirectional) * 3 (encoders)

## References


[1] T. H. Nguyen and R. Grishman, "Relation extraction: Perspective from convolutional neural networks." in *VS@ HLT-NAACL*, 2015, pp. 39–48.

[2] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, "Neural relation extraction with selective attention over instances." in *ACL (1)*, 2016.

[3] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[4] D. Zhang and D. Wang, "Relation classification: Cnn or rnn?" in *International Conference on Computer Processing of Oriental Languages*. Springer, 2016, pp. 665–675.

[5] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," *The semantic web*, pp. 722–735, 2007.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," in *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne, 181*, vol. 185, 1997.

[8] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[9] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager *et al.*, "Building watson: An overview of the deepqa project," *AI magazine*, vol. 31, no. 3, pp. 59–79, 2010.

[10] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pp. 753–753, 2005.

[11] E. S. Soofi, "Principal information theoretic approaches," *Journal of the American Statistical Association*, vol. 95, no. 452, pp. 1349–1353, 2000.

[12] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ICLR*, 2014.

[13] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[14] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.