On the Recursive Neural Networks for Relation Extraction and Entity Recognition

Daniel Khashabi
Computer Science Department
University of Illinois, Urbana-Champaign
Urbana, IL 61801 USA
khashab2@illinois.edu

Abstract

Recently there has been a surge of interest in neural architectures for complex structured learning tasks. Along this track, we are addressing the supervised task of relation extraction and named-entity recognition via recursive neural structures and deep unsupervised feature learning. Our models are inspired by several recent works in deep learning for natural language. We have extended the previous models, and evaluated them in various scenarios, for relation extraction and named-entity recognition. In the models, we avoid using any external features, so as to investigate the power of representation instead of feature engineering. We implement the models and proposed some more general models for future work. We will briefly review previous works on deep learning and give a brief overview of recent progresses relation extraction and named-entity recognition.

1 Introduction

The problem of relation extraction has a crucial impact in a lot of NLP applications, from information extraction to entailment entailment. In this problem, the goal is to determine if the text expresses a semantic relation between two entities, if so, what relation is expressed. The goal is thus twofold: identifying named-entities in text, and determining the semantic relation between them. This is mostly the traditional view of relation extraction which is still being used widely, while in Open Information Extraction view of relation extraction the relations are not known a priori. The difficulty in extracting relations is that, generally they are not explicitly mentioned, but it could be inferred from semantic connection between named-entities, and the structure of this connection can be highly complicated and ambiguous, even for human. There are different sources of ambiguity. One type of ambiguity can be because of the grammatical structure, for example consider the sentence *"Newton was a fellow of Trinity College, Cambridge"*, in which we can see the relationship `Person-Affiliation (Newton,Trinity College)` clearly conveyed by the sentence. In addition we can see the implicit relations `Located-at(Trinity College, Cambridge)` and `Person-Affiliation (Newton,Cambridge University)`. The *synonymy* is another source of ambiguity, for example in sentence *"Anna runs the company"*, we see the relationship `Person-Affiliation (Anna,the company)`, while in the sentence *"Anna runs upstairs"* there is no relationship, though having very similar structure to the previous sentence. There is an interesting discussion of ambiguity in natural language could be found at (Al Fawareh et al., 2008).

In the recent years, neural models have been repopularized, due to recent innovations in deep learning, automatic feature learning and model representation. Traditionally researchers try to improve the performance of their systems using feature engineering. For example (Klein and Manning, 2003) by using fine-grained, goal-oriented

Technical Report(May, 2013). This is mainly done for CS546 at UIUC, under the supervision of Prof. Dan Roth.

features that capture more syntactic and semantics has considerably increased the performance of parsing. (Bengtson and Roth, 2008) and (Ratinov and Roth, 2009) are also good examples in which features and their effect on the performance results are carefully studied. While in deep learning, the goal is to tackle the feature representation by creating a big meaningful network that is able to capture the features in the structured data. Thus, some part of representation(features) could be learnt from abundant unsupervised data, and partly from supervised data, while learning the main task.

This document includes our results for supervised relation extraction and named-entity recognition via neural structures. We use ACE-2004 data for both named-entity and relation extraction. Our models are inspired by (Socher et al., 2012) which recently showed the power of their compositional operators for relation classification. We are extending the proposed model in several ways to see the effectiveness of the model. In none of the models discussed here, we use any external features, so as to avoid feature engineering. The only external tool used, is syntactic parse of each sentence, as it is the case in (Socher et al., 2012). We tested the models and showed that the results are comparable with state-of-the-art results. We also proposed some more models for future work. Section 2 briefly covers review of some recent ideas for implementing deep model in NLP. Section 3 reviews some important related works for relation extraction and named-entity recognition. Section 4 includes our models, results and some analysis.

2 Related works on deep (neural) language learning

The main incentive for deep learning is to automate feature learning during unsupervised pre-training deep network of variables, instead of hand-designing them. One of the tools that made it possible to do numerical operations on words based on their meanings, is their vector-space representation in (Bengio et al., 2003; Turian et al., 2010). By replacing words with vectors and pre-training on big set of unlabelled data one can automatically learn the needed representation for the target task. Having the vector-space representation, there can be many different models to could combine words, train the parameters of the model, and make sensible decisions based on them. A group of works (Collobert and Weston, 2008; Weston et al., 2008; Collobert et al., 2011) investigate the properties of convolutional neural networks for sequence labelling tasks. Their results are highly competitive or in some cases superior than the state-of-the-art algorithms. Using their model, they show how to do multi-task learning, i.e. learning and inference on several tasks, at the same time, though in many cases multi-task learning does not increase the overall performance of the model. Another inspiring point in their work is that, their lookup table is trained with millions of sentences in an unsupervised scheme, on a feed forward network using an n-grams of a corpus as a positive example and corrupted the last word of the n-gram as a negative example. There are also some similar works in (Mnih and Hinton, 2009; Bengio et al., 2009; Mnih and Teh, 2012; Bordes et al., 2012).

Though the aforementioned models have very nice results, one big drawback in their works is that, the models don't have the capacity to include compositional information of natural language into itself. Recursive Neural Networks(RNN), which give the ability to learn compositional structures, have widely been used in areas like Computer Vision, and NLP recently(Socher et al., 2010). In an RNN, two inputs, which are usually an equivalent embeddings, are given to a network, which has an output of the same dimensions as its input. This procedure could be repeated for many times until we are left with only one output (e.g. n-dimensional embedding output for the whole sentence) based on which one can perform decision-making. For n-dimensional embedding, to get output of the same dimension as each of the inputs, one can do the non-linear operation, e.g. $p = f(W[c_1; c_2] + b)$ where $W \in \mathbb{R}^{n \times 2n}$ is the weight vector that needs to be trained and $c_1, c_2 \in \mathbb{R}^n$ are the input vectors to the network which are given by the pre-training network. Such a recursive network could be trained by calculating errors at each vertex and propagating it through the structure (Goller and Kuchler, 1996) and updating parameters for both the network

and the input word embeddings. To get different variations of RNNs one could devise different operations on the inputs. For example, in (Socher et al., 2012) instead of using only vector-space representation for words, additional matrix for each word is added to make sensible compositionality in language, and the model is called *Matrix-Vector* Recursive Neural Network (MV-RNN). They apply this for relation classifications on syntactic parse tree of the sentences, spanned from the first mention to the second mention.

In addition to modelling, deep learning methods also differ in their unsupervised pre-training strategy. A group of works like (Collobert et al., 2011) perform the pre-training on the whole network, while (Socher et al., 2011) does this layer-wise using a recursive autoencoders on word vectors, i.e. at each node, after calculating the output of the non-linear function, they feed it to a reconstruction layer, and minimize the reconstruction error. In another similar scheme, instead of minimizing reconstruction error at each node, the whole tree is reconstructed and error minimization is applied to each leaf node.

3 Related works on relation/entity extraction

In the traditional view of relation extraction, usually some textual data is annotated according to predefined protocol for semantic relation between entities, and this labelling is then used in supervised training. While in Open Information Extraction the relations' protocol is not defined a priori(Banko et al., 2009). Our work lies in the traditional track of the works.

There are various paradigms applied to relation extraction. In (Sarawagi, 2008) there is a very comprehensive review of the past works. Most of the previous works are *supervised* approaches, i.e. they used hand-labelled corpus to learn the relation patterns between words (Culotta and Sorensen, 2004). Most systems, extract lots of lexical, syntactic and semantic features, by which they predict the relation between given pair of entities. There are several problems with these approaches. First, having labelled data is expensive and limited; therefore researchers look for different ways that they could get rid of limiting themselves to labelled data. Another deficiency is feature extraction; for many applications, hand-engineering features is sometimes very tricky and laborious task; often it needs a lot of tests to find the most informative features and removing redundant information. Another problem is that, since the classifiers are trained on a limited data, they are biased towards that corpus, and do not perform very well outside that domain.

To compensate for lack of hand-labelled data, a group of works are purely *unsupervised*, i.e. do not use any labelled data. In fact, they only use the syntactic and lexical patterns in the given text, to cluster relations. These models are able to process very big corpus, while the results might lack semantic coherence. Such models could be found in (Shinyama and Sekine, 2006; Banko et al., 2009). Another approach is to use small number of data (comparing to size of unsupervised data) to extract new set of patterns, and extract more instances, and in turn use them to extract more patterns; in other words they are *bootstrapping* the seed information by reusing them several times (Bunescu and Mooney, 2005). This work is using result of pre-trained unsupervised word embeddings, and trains them in a supervised scheme, thus benefits from both labelled and unlabelled data.

The limitations in the previous works, resulted in another paradigm commonly known as *distant supervision* that use big database of entities connected to each other, like Freebase (Mintz et al., 2009; Riedel et al., 2010), which aligns the database records with the sentences that include information about these records. Learning in this big network, somehow eliminates the problem of overfitting to corpus, as it was the case in *supervised* paradigm.

Having named-entity labels is very informative for finding the relation type between them, and vice versa having the relation type between words, ease problem of named-entity tagging. Some works have simplified the problem by pipe-lining one algorithm with the other one, while in reality their results are

interdependent. (Finkel et al., 2006) shows improvements by pipe-lining M-best solutions in a probabilistic model. (Roth and Yih, 2007) simplifies the joint relation extraction and named-entity recognition, to separate training time, and inference by modelling constraints. In this work, we model the problem both in pipeline and jointly.

4 Models and results

Named-entity recognition is classifying words in predefined categories that generally have outer real embodiment. We selected a big subset of the ACE-2004 corpus which includes about 26,025 named-entity mentions, and 5,768 relation instances, from 6,102 sentences. For relations we have 23 classes and for named-entities, there are 7 classes. The classes are explained in the appendix, in Table 10 and Table 9 at the end of this document. Relations are limited to any two named-entities at the same sentence.

In some of the tasks we assume having a `Null` class when we don't want to classify the target (entity or relation) in one of the predefined categories. Considering a `Null` class in the classification, most of candidates (more than 90% both in entity recognition and relation extraction) lie in the `Null` class. Thus, training a model with all `Null` samples, there might create a bias towards the `Null` class. Thus one might consider a pipeline scenario during which, first a binary classifier distinguishes between `Null` and non-`Null` items, and then passes the result to a second classifier for a fine classification. For each of scenarios we calculate average accuracy, precision, recall and F1 for classification.

First for any word we assume having an assigned embedding in continuous n-dimensional vector-space. This might be seen as a pipeline, but in practice the words vectors are used as prior information which are being modified during the training procedure. For this representation, we used the embeddings trained by (Turian et al., 2010) which is inspired by (Collobert and Weston, 2008) in a lookup table of dimensionality $\mathbb{R}^{|V|\times n}$ where n is the size of each word vector we defined and

`http://projects.ldc.upenn.edu/ace/data/`

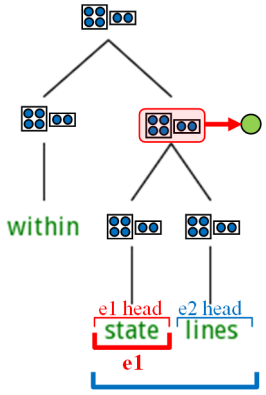

Figure 1: The model for "Mention $\xrightarrow{\text{MV-RNN}}$ (non-null)Relation Types", via considering only head words for each mention. The head words are the most important words in each mention and are given by the dataset.

V is the set of all words in our vocabulary. Turian provided several embedding matrices for different n, from 25, 50 or 100; here we used $n = 50$.

In models discussed here, we have a fixed structure for a neural structure, which has the same input/output dimensionality, uses previous output as its input signal, which allows it to remember the inputs over time (or over structure); thus called Recursive Neural Networks(RNN). The recursions are done on the parse tree of sentences using Stanford parser (Klein and Manning, 2003). Following exactly (Socher et al., 2012) we are using RNN based on Matrix-Vector(MV-RNN) scheme, which is considering one relation per sentence, and mentions are disjoint. We use the same operators for the combination of the word-vectors and optimization of the word operators and word vectors as explained in (Socher et al., 2012), but we extend the model in various ways. Here we have divided the problem into several stages, and several submodels. In the following parts, we explain our models, and the results we got from them.

4.1 Mention $\xrightarrow{\text{MV-RNN}}$ (non-null)Relation Types

In our preliminary model, we only assume the first word of the "head words" for each mentions, and the problem becomes identical to the one solved in (Socher et al., 2012). The "head words" are

`http://metaoptimize.com/projects/wordreprs/`

	only mention heads	with mention sub-trees
Avg. Accuracy %	69.01	71.24
Avg. Precision %	63.19	70.18
Avg. Recall %	57.62	62.00
Avg. F1 %	59.33	64.45

Table 1: Results for "Mentions $\xrightarrow{\text{MV-RNN}}$ (non-null)Relation Types"; results on the test data, considering only non-Null relations for training and testing. The results are average on 5-fold cross validation, and trained on 5,768 relation instances.

directly given in the ACE dataset. One sample phrase and recursive decomposition, based on its syntactic parse tree is depicted in Figure 1. As shown, first, we find the head words for the first and second mentions and recursively calculate the matrix-vector representation on the whole tree that spans the two mention heads. Then classify the relation type, using the output vector. To make the classification more accurate, the depth of the tree, distance between entities, two inner and outer words are also included in the output prediction vector, when they are possible. We first assume that we are classifying all of the non-Null relations. In other words, while training and inference, we only assume any two mentions, that are surely "related", and give them to the relation classifier. This is sometimes called *relation classification*, in contrast to *relation extraction*. The results of such classification is shown in Table 1 under "only relation heads" column. The results are averaged over 5-fold cross-validation, and trained on 5,768 relation instances. Since the mention heads are not enough to make predictions in some cases, and to better capture information about each of the named-entities, we consider full tree for each mention, we calculate the recursive vector-matrix values for each mention, in addition to the whole tree that includes the two mentions. A visualization of this model is depicted in Figure 2. The improvement of this modelling is shown in Table 1 under "with mention sub-trees" column. This gives a considerable improvement over the "only mention heads" model. Note that this model can now handle relations between two nested named-entities.

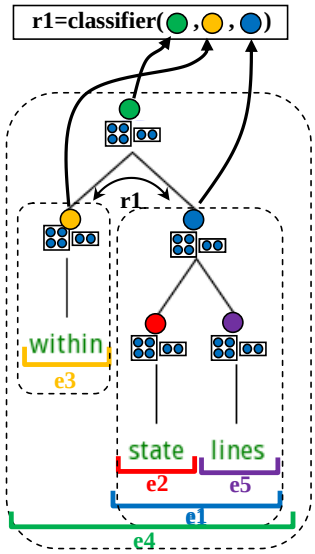

Figure 2: The model for "Mentions $\xrightarrow{\text{MV-RNN}}$ (non-null)Relation Types"; with considering the sub-trees spanning each of the mentions.

We can add more features to the classification, e.g. mention types (NER), part-of-speech-tags(POS), Wordnet hypernyms(HYP), etc just like what is considered at (Socher et al., 2012). Here we avoid doing so, since we only want to focus on the performance of the neural model, regardless of any external features.

4.2 Mention Heads $\xrightarrow{\text{MV-RNN}}$ Binary null/non-null relation classes

For extracting relations, given only mentions, we need to classify their relation, into either Null, or other valid relation types. In our database, there are 285462 pair of entities that are not related(or classified as Null relation type), based on ACE-2004 annotation; while number of related pairs is 5,768. Therefore the number of the Null relations is almost 50 times more than the whole number of instances for non-Null relations, or in other words, more than 97% of the whole data lies in this class. Thus, training a model with the Null samples, might create a bias towards the Null class. Thus one might consider a *pipeline* scenario during which first a binary classifier distinguishes between Null and non-Null items, and passes the result to a second classifier for a fine classification. Results of

# of Null instances / # of non-Null instances	≈ 1		≈ 2		≈ 5		≈ 49.5	
Avg. Accuracy (%)	83.6	83.9	87	84.9	91	88.7	98	97.9
Avg. Precision (%)	83.5	83.9	83.5	83.1	80.9	80.3	72.5	68.8
Avg. Recall (%)	83.5	83.9	82.7	83.0	77.0	80.4	55.1	53.2
Avg. F1 (%)	83.5	83.9	83.1	83.0	78.7	80.1	58.0	55.2

Table 2: Results for "Mentions $\xrightarrow{\text{MV-RNN}}$ Binary null/non-null relation classifier". The left numbers belong to the model only with mention heads, and the right numbers belong to the model with mention subtrees.

	cascaded classifiers	+null classifier
Accuracy (%)	37.9	98.0
Precision (%)	37.3	71.2
Recall (%)	32.9	23.3
F1 (%)	34.2	25.8

Table 3: Results of classification of relation, cascaded classifiers, and one classifier, which also includes Null type.

the binary non-Null/Null classification are summarized in Table 2.

4.3 Mention Heads $\xrightarrow{\text{MV-RNN}}$ (+null)Relation Types

In this model, instead of pipelining, we want to train one classifier, for all 24 classes (23 relations + one Null class). The inputs to the model are mention heads, thus we call this mode, Mention Heads $\xrightarrow{\text{MV-RNN}}$ (+null)Relation Types. Putting (+null) is to emphasize that, the model does include the Null class. The results of this classification is summarized in Table 3, under "+null classifier" column. Note that in this setting we choose the number of Null instances in the training, to be as big as our instances for other relations. In the table, the results of cascading two classifiers is also included. Interestingly the "+null classifier" gives better F1 score than "cascaded classifiers".

The results summarized in the Table 3, as mentioned in the last paragraph, are trained on a corpus in which the number of Null instances, is almost the same as the number of instances for other relations. The results are also tested on the same corpus. This might be subject to argument, since in practice if we want to test a system, on real data, in most of the cases(may be more than 95 percent) we face a Null relation. Thus, one could argue that, we should test the system, with more Null/non-Null ratios.

4.4 Mentions $\xrightarrow{\text{MV-RNN}}$ (non-null) Named-Entity Types

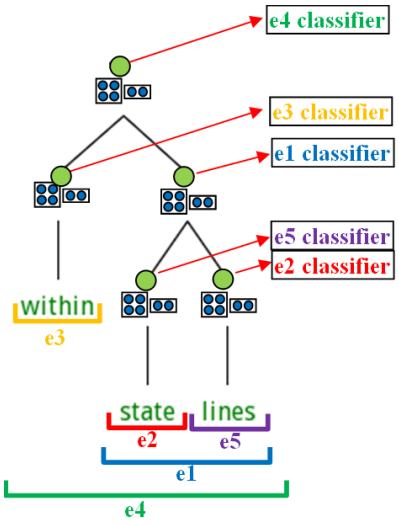

Figure 3: The model for "Mentions $\xrightarrow{\text{MV-RNN}}$ (non-null) Named-Entity Types"; classification of entities is being done by recursively calculating matrix-vector values at root vertices and doing prediction about the sub-tree using a softmax classifier on the root vector.

Using a similar model, we want to predict entity types for a bunch of adjacent words. For each entity mention, we consider the parse tree that spans it, and classify using the calculated matrix-value at the root vertex with a softmax classifier. The structure of the model is depicted in Figure 3, where some root vertices are depicted how to use for classification.
In this part, we assume that, in the prediction, we don't include the Null named-entity. The results are listed in the Table 4.

Avg. Accuracy %	87
Avg. Precision %	83.86
Avg. Recall %	76.69
Avg. F1 %	79.93

Table 4: Results for "Mentions $\xrightarrow{\text{MV-RNN}}$ (non-null) Named-Entity Types"; results on the test data, considering only non-Null entities for training and testing. The results are average on 5-fold cross validation, and trained on 26,025 entity instances.

4.5 Plain Text $\xrightarrow{\text{MV-RNN}}$ Binary (null/non-null) Named Entities

This is similar to the previous model, with the difference that, we might be given Null entities in the input. In other words, we can assume that, we are given plain text without any annotation. Similar to the previous ones, we generate the parse tree for each sentence, as depicted in Figure 3. Assuming that, all the words in one named-entity, are adjacent (which is the case, at least in ACE-2004), at any node, we use the trained classifier to classify the set of words under the that node. Similar to the previous problems, many of the given instances are Null; so we try several settings for testing this model.

In this model, we assume that, words are from the same named-entity, only if they share a common parent that spans both of them, and does not span any other words. For example, in Figure 3, $e2$ and $e5$ could create a named-entity, because they share the blue vertex which is spanning both of them, and does not span any other words. But $e3$ and $e2$ cannot create a named-entity since their only common parent (the green vertex) covers another words, i.e. $e5$. Also $e1$ and $e3$ could create a joint named-entity, since they share a parent (the green vertex) which spans both of them, and not any redundant words. Note that, based on this assumption, it is trivial that, words should be adjacent to each other(not separate blocks of words).

When seeing a candidate instance, we first check whether its words are adjacent in the main sentence, or not. If so, we then check whether they share a common root in their parse tree, whether it includes the candidate and not any other words. If it is also passes the previous test, we give the candidate to our classifier. Thus the classifier is only needed to be trained on valid candidates. To generate valid Null instances for the classifier, we can just get sentences, parse them, and at each vertex take the whole spanning leaves as one potential named-entity mention. One can limit the length of candidates here, since the width of parse trees can become so big. In our experiments we limited the maximum width to 10 words. By following this process, we generated 234,319 Null instances, along with 26,025 non-Null instances (almost 10 times). Using this syntesized training set, we can train our classifier with a chosen ratio for the number of the Null instances to non-Null instances. When this ratio is zero, it is like when all mentions are given and we only want to classify them into non-Null classes. When the ratio is around 10, it is like looking at a plain text without any knowledge given. Results are summarized in Table 5.

$\frac{\text{\# of Null instances}}{\text{\# of non-Null instances}}$	≈ 1
Avg. Accuracy (%)	87
Avg. Precision (%)	86.24
Avg. Recall (%)	82.46
Avg. F1 (%)	84.39

Table 5: Results for "Plain Text $\xrightarrow{\text{MV-RNN}}$ (null/non-null)Named Entities".

$\frac{\text{\# of Null instances}}{\text{\# of non-Null instances}}$	≈ 1	≈ 2	≈ 3	≈ 10
Avg. Accuracy (%)	81	84	85	89
Avg. Precision (%)	76.31	72.6	73.3	68.4
Avg. Recall (%)	58.90	58.20	55.0	37.5
Avg. F1 (%)	64.03	62.02	58.7	37.6

Table 6: Results for "Plain Text $\xrightarrow{\text{MV-RNN}}$ (+null) Named Entities"

4.6 Plain Text $\xrightarrow{\text{MV-RNN}}$ (+null) Named Entities

Instead of pipelining two classifiers, we could use a bigger classifier classifier for named-entity resolution. In the Table 6 we have summarized our results for different rations of Null instances to non-Null instances.

In the Table 7 results of pipelining two classifiers is summarized.

$\frac{\text{\# of Null instances}}{\text{\# of non-Null instances}}$	≈ 1
Avg. Accuracy (%)	75.29
Avg. Precision (%)	72.32
Avg. Recall (%)	65.27
Avg. F1 (%)	68.28

Table 7: Result of pipeling two classifier.

4.7 Plain Text $\xrightarrow{\text{MV-RNN}}$ (+null) Relation Types

Now we go back to the last relation extraction model, "Plain Text $\xrightarrow{\text{MV-RNN}}$ (+null) Relation Types" and extend it more. Since in many cases having named-entity mentions in text demands having a preprocessing layer for finding the valid mentions, we directly target a plain text, and find the relations between possible combinations of words. As mentioned before, in all of the tasks and modellings we consider having the parse tree of sentences, which is not a big assumption. For generation of possible candidates we follow a similar path that we followed at Section 4.5 for generation of all of the named-entity candidates in plain text using its parse tree. Having the named-entity candidates, we create the relation candidates by choosing any two named-entity candidates at the same sentence.

4.8 Future works

Here we summarize a set of more ideas for future work.

4.8.1 Joint learning/inference

In addition to fully supervised models, there are many other sources of information one can exploit, e.g. unlabelled data, knowledge bases, real-world facts, transfer learning (cross-knowledge between different tasks). Such transfer might transfer learning comes from appropriate joint learning of several tasks. Doing joint learning of several tasks in an efficient way, has always been a big dream. One of the recent works in implementing joint learning and inference is shown in (Collobert et al., 2011), however their joint model does not increase their overall performance considerably(and in some cases the performance decreases). The speculation whether joint learning/inference will help the results, is still under debate; since the effectiveness of joint learning/inference is mostly a matter of choosing the right modelling and representation.

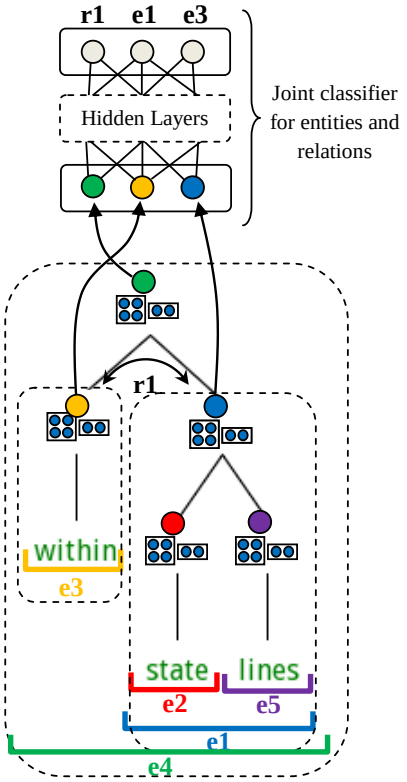

Figure 4: Joint classification of entities and relations, with several sub-trees.

Regarding neural networks' capabilities in multi-output learning, investigating its capabilities in joint learning NLP task is definitely an interesting issue. An example of such joint learning, could be learning named-entity recognition and their relationships. Assume that we are given plain text and we want to find any named-entity and any relation among named-entities. A model for this task is depicted in Figure 4.

4.8.2 ILP-based inference for hard-constraints

Though such a joint model might have a very important capabilities for transferring information between several tasks, it is never enough. In fact, it might make very ridiculous mistakes. In addition to training data annotation, there are a lot of information that come from the connections between entities in the real world, but are not directly encoded in the training data; for example the *fact* that a "person" cannot be located at a "person" is a hard constraint which has roots in our real world. There are many previous works that target incorporation of back-

$R \in \{1,2,3,23\}$	$E_1 \in All$	$E_2 \in All$
$R \in \{4,5,6\}$	$E_1 \in \{b\}$	$E_2 \in \{b\}$
$R \in \{7,8,...,13\}$	$E_1 \in \{1,2,7\}$	$E_2 \in \{1,2,7\}$
$R \in \{14,15,16\}$	$E_1 \in \{1,2,7\}$	$E_2 \in \{1,3,4,5\}$
$R \in \{17,18,19\}$	$E_1 \in \{1,2,6,7\}$	$E_2 \in \{1,2,6,7\}$
$R \in \{20,21\}$	$E_1 \in \{1,2,7\}$	$E_2 \in \{1,6\}$
$R \in \{22\}$	$E_1 \in \{1,2,6,7\}$	$E_2 \in \{1,6\}$

Table 8: The set of entity-relation constraints; in the table $All = \{a, b, c, d, e, f, g\}$. The numbers and alphabets are based on the order shown in Table 10 and Table 9.

ground knowledge, mostly as hard constraints(Chan and Roth, 2010). A relatively more complete version of the constraints studied at (Chan and Roth, 2010) is shown in Table 8.

The constraints could be applied to the problem in different ways. The most simple case is to train the classifier for both entities and relations *independently* and apply the constraints when test-time decision inference, i.e. choose the most-probable answer which satisfies the constraints via ILP. In addition to the test time, one might consider applying the ILP corrections during training time, while doing back propagation of the RNN.

1.**Forward propagation** of beliefs in a neural network
2.**Joint inference correction** via ILP among all output variables
3.**Backpropagation** of prediction-errors along the neural network.

5 Conclusion

In this project, we designed and implemented several neural models for relation and entity recognition. We built our models based on several previous works on recursive neural language processing. We showed how to formalize different problems to feed them into the neural structure. We implemented some of these structures and showed effectiveness of their results. Though our approach here was mostly supervised, the appealing points in our implementations is that, we are not using any external features here, and still getting good results. However it is important to mention that, the results can definitely be improved by further addition of features.

One of the tasks that is interlocked with named-entity recognition and relation extraction, is the *coreference resolution* problem, in which is the goal is to find find and link mentions in different sentences (documents) that point to the same object. The challenging part is that, objects might be related in a very complex ways that demand semantic understanding of sentences. It would be interesting to investigate more informative operators for composition of words, that could give better results on complex tasks like coreference resolution. Another issue is the alignment of unsupervised pre-training, with supervised tasks. Though we used previously trained vectors, but the unsupervised and supervised stages, were not aligned. In other words, the unsupervised phase was not intended for relation-extraction/named-entity recognition tasks. That's why, it might not be as good as it should. It would be interesting to investigate ways to induce *weak* supervision, and *indirect* supervision for neural architectures.

ILP: Integer Linear Programming

6 Acknowledgement

Thanks to Richard Socher for his comments. Also in the implementation of the models discussed here, we have used the code published at (Socher et al., 2012) which is available at `www.socher.org` and is originally written for SemEval-2010 relation classification task. Also thanks to Shalmoli Gupta for providing her list of relation/entity constraints.

References

H.M. Al Fawareh, S. Jusoh, and W.R.S. Osman. 2008. Ambiguity in text mining. In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 1172–1176.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2009. *Open information extraction for the web*. University of Washington.

Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *JOURNAL OF MACHINE LEARNING RESEARCH*, 3:1137–1155.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in*

Natural Language Processing, pages 294–303. Association for Computational Linguistics.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 152–160. Association for Computational Linguistics.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 423. Association for Computational Linguistics.

Jenny Rose Finkel, Christopher D Manning, and Andrew Y Ng. 2006. Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 618–626. Association for Computational Linguistics.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. 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*, pages 1003–1011. Association for Computational Linguistics.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. *Advances in neural information processing systems*, 21:1081–1088.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. *Introduction to Statistical Relational Learning*, pages 553–580.

Sunita Sarawagi. 2008. Information extraction. *Foundations and trends in databases*, 1(3):261–377.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 304–311. Association for Computational Linguistics.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method

#	Relation Class Label
1	PHYS:Located,
2	PHYS:Near,
3	PHYS:Part-Whole,
4	PER-SOC:Business,
5	PER-SOC:Family,
6	PER-SOC:Other,
7	EMP-ORG:Employ-Executive,
8	EMP-ORG:Employ-Staff,
9	EMP-ORG:Employ-Undetermined,
10	EMP-ORG:Member-of-Group,
11	EMP-ORG:Subsidiary,
12	EMP-ORG:Partner,
13	EMP-ORG:Other,
14	ART:User-or-Owner,
15	ART:Inventor-or-Manufacturer,
16	ART:Other,
17	OTHER-AFF:Ethnic,
18	OTHER-AFF:Ideology,
19	OTHER-AFF:Other,
20	GPE-AFF:Citizen-or-Resident
21	GPE-AFF:Based-In,
22	GPE-AFF:Other,
23	DISC,

Table 9: 23 relation types used in our system.

for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jason Weston, Frédéric Ratle, and Ronan Collobert. 2008. Deep learning via semi-supervised embedding. In *Proceedings of the 25th international conference on Machine learning*, pages 1168–1175. ACM.

Appendix: Relations, and Entities in ACE-2004 data

Relations are defined between two named-entities. For example, in the following sentence,

`[Microsoft]`$_{e1}$ `spokesman, [Bob Jones]`$_{e2}$`.`

`[Microsoft]`$_{e1}$ and `[Bob Jones]`$_{e2}$ are first and second named entities, with `entity-type(Microsoft) = ORG` and `entity-type(Bob Jones) = PER`, respectively. The relationship between $e1$ and $e2$ is `relation-type(Microsoft, Bob Jones) = EMP-ORG:Employ-Undetermined`. The entities could be inside each other; for example,

`The state had trouble keeping people down within [[state]`$_{e1}$ `lines]`$_{e2}$`.`

In the above sentence, `entity-type(state ) = GPE`, `entity-type(state lines) = LOC` and `relation-type(Microsoft, Bob Jones) = PHYS:Near`.

Based on the official documentation, there are following 23 relation types which are summarized in Table 9. The entity categories are also summarized in Table 10.

#	Entity Class Label	Explanation
a	**GPE**(Geo-political Entity)	Limited to geographical regions defined by political and/or social groups. A GPE entity subsumes and does not distinguish between a nation, its region, its government, or its people.
b	**PER**(Person)	Limited to humans, single or plural.
c	**FAC**(Facility)	Limited to buildings, real estates, and permanent man-made structures.
d	**WEA**(Weapon)	Limited to physical devices primarily used as instruments for physically harming or destroying animals (often humans), buildings, or other constructions.
e	**VEH**(Vehicle)	Limited to physical device primarily designed to move an object from one location to another.
f	**LOC**(Locations)	Limited to geographical areas and landmasses, bodies of water, and geological formations.
g	**ORG**(Organization)	Limited to corporations, agencies, established groups or organizational structure

Table 10: 7 entiity classes used in our system, and their explanations.