Semantic Loss Application to Entity Relation Recognition

Venkata Sasank Pagolu[1]

***Abstract*— Usually, entity relation recognition systems either use a pipe-lined model that treats the entity tagging and relation identification as separate tasks or a joint model that simultaneously identifies the relation and entities. This paper compares these two general approaches for the entity relation recognition. State-of-the-art entity relation recognition systems are built using deep recurrent neural networks which often does not capture the symbolic knowledge or the logical constraints in the problem. The main contribution of this paper is an end-to-end neural model for joint entity relation extraction which incorporates a novel loss function. This novel loss function encodes the constraint information in the problem to guide the model training effectively. We show that addition of this loss function to the existing typical loss functions has a positive impact over the performance of the models. This model is truly end-to-end, requires no feature engineering and easily extensible. Extensive experimentation has been conducted to evaluate the significance of capturing symbolic knowledge for natural language understanding. Models using this loss function are observed to be outperforming their counterparts and converging faster. Experimental results in this work suggest the use of this methodology for other language understanding applications.**

I. INTRODUCTION

Entity relation recognition is one of the key tasks in natural language understanding which is beneficial to many other tasks such as Question Answering and Knowledge base population. The approaches that are used to solve this problem can be divided into two categories 1) pipeline approaches, treating entity labeling and relation extraction tasks separately 2) joint entity and relation extraction. It is argued in the previous works [1] that the joint approaches are beneficial compared to the pipeline approaches. The pipeline approaches face issues with loss propagation between two tasks. For example, knowing that the relation is kill, it is easier to recognize the two entities involved as of person category. These two tasks are mutually cooperative as one increases the confidence of prediction in another task. We have built models and conducted experiments to compare results from both the approaches. State-of-the-art joint models are built using recurrent neural networks to encode the contextual information in the input sequence. The current state-of-the-art models [3], [1], [5] for joint entity relation extraction are a combination of Bidirectional LSTM units and NLP techniques. [3] used LSTM networks for entity identification and a separate tree-based LSTM network for relation extraction. So, their model was heavily dependent on the accuracy of dependency parser to generate parse trees. [1] solved this problem without dependency trees using attention in the LSTM model. The drawback of this approach is that it assumes the mutual exclusion of a word in relation classes. [5] solved this drawback by treating it as a multihead selection problem and jointly extracted entities and relations.

All of these approaches lack a mechanism that captures the symbolic knowledge that the problem possess. A relation like kill can only exist between two entities that are of type person. A relation like birthplace is only between a person entity and a location entity. The primary motivation behind our work is to effectively capture the symbolic knowledge in the form of Boolean logic constraints to achieve a guided training and learning of the model. With these additional constraints in the form a loss function, the convergence of the model could be achieved in less number of epochs and the model performance could be enhanced. This work does extensive experimentation to realize the importance of constraint formulated loss for entity relation recognition.

Semantic Loss proposed in [2] is an efficient approach to augment the deep learning approaches with symbolic knowledge. It is a measure of how good the network is in satisfying the output constraints. We have used the semantic loss to improve the performance of our joint entity relation extraction model. We have also incorporated semantic loss in our named entity recognition model to evaluate the significance of this loss function for one hot encoding type constraints. The contributions of our work are twofold. 1) Developed efficient models for joint entity relation extraction, 2) Evaluated the importance of symbolic knowledge for natural language understanding tasks.

The rest of this paper is organized as follows. In section 2, we present the previous works in this domain followed by the description of statistics of the dataset used in this work in section 3. Section 4 details the methodology adopted including the model architectures and their variations. We present our experimental results and discussion in section 5 followed by conclusion in section 6.

II. RELATED WORK

The problem of joint entity relation extraction has been studied extensively in the past. There are broadly two different approaches that are taken to solve this problem. First, the problem is divided into two sub problems, Named Entity Recognition and Relation Extraction, each to be solved separately. Second approach is to jointly learn and extract the entities by treating it as an end-to-end application.

A. Named Entity Recognition

Named Entity Recognition (NER) is the first task to be solved in the end-to-end relation extraction problem. Earlier

[1] Computer Science Department, University of California Los Angeles vpagolu@cs.ucla.edu

approaches to solve this problem relied heavily on hand-crafted features using Conditional Random Fields [6], Maximum Margin Markov Networks [7] Support Vector Machines for structured output [8]. [9] used Hidden Markov Models to tag the entities of words in the sequence.

Recently, neural approaches have been performing well on the publicly available NER datasets. [4] have proposed a neural model that consists of a Bidirectional LSTM to extract word level embeddings, a convolutional neural network to obtain character level representations of words and these two representations are combined and processed with a CRF layer to find the most probable entity type for every word in the sentence. This architecture solves the problem by treating it as end-to-end without the help of hand engineered features. [10] have also proposed a neural architecture like the one in [4] except that their architecture does not use a convolutional neural network for extracting character level representations. They have also proposed a transition based chunking model that treats input as a sequence of chunks and labels them using an algorithm similar to transition based dependency parsing. These methods achieve state-of-the-art performance on NER problem.

B. Relation Extraction

Relation extraction is the second step in this task. Similar to NER, relation extraction is also used to be solved with the help of hand-crafted features. [11] has employed maximum entropy models to combine the extracted lexical, syntactic and semantic features from the text. [12] pointed out the problems with long range dependencies using feature-based techniques and proposed kernel-based evaluation for extracting relations. They have defined kernels over the text representations and used them in conjunction with support vector machines and voted perceptron algorithms to extract relations. Recently, neural approaches are outperforming previous approaches for this task. [13] have proposed a Convolutional Neural Network combined with a pairwise ranking loss function to extract relation between the target nominals. They showed that their approach outperforms the hand-crafted feature-based approaches and this is more effective than the CNN with a softmax. [14] have proposed a recurrent neural network which is an LSTM coupled with the shortest dependency path information between two entities. They have also used a custom dropout strategy to avert the problem of overfitting. [15] proposed a novel neural model that is a combination of both the convolutional and recurrent neural networks which is trained by a simple voting scheme. This model achieved the state-of-the-art performance for the relation classification problem. All these approaches have performed well and eliminated the need to use the hand-crafted features.

C. Joint Entity and Relation Extraction

Most of the approaches that identify the entities and relations among them jointly rely heavily on the external NLP tools such as POS taggers, dependency parsers etc. [16] have proposed a table-based representation of entities and relations and their approach is based on an inexact search on the table. They have used various feature engineering techniques to solve the joint entity relation extraction. The problem with these approaches are that they do not scale well to novel applications, languages and also increase the computational complexity.

Recently, neural approaches are well studied and observed to outperform the feature-based approaches for joint entity-relation extraction. [3] proposed a neural model that is a stacking of bidirectional tree structured LSTM on bidirectional sequential LSTM. The purpose of the model is to jointly extract the word level information along with the dependency tree structure information. [17] solved this problem using globally normalized convolutional networks. They used a convolutional neural network in conjunction with a linear chain conditional random field to predict the sequence of entities and relations among them at the same time. Their experiments revealed that global normalization is better than local normalization using a softmax. [1] have proposed a novel neural architecture based on incremental learning and embeddings of labels for joint entity relation extraction. Their model uses an attention-based LSTM which does not need dependency tree structure information. Their model also uses pointer generator networks for determining relations between entities. [5] defined this as a multihead selection problem using a sigmoid loss to obtain multiple relations that are not mutually exclusive. They used a CRF loss for the NER component. These neural approaches revealed that the joint learning for entity and relations is better than the pipelining approaches. None of the previous works have encoded the problem constraint knowledge into a loss function. We have taken this approach to verify the symbolic knowledge usefulness to natural language understanding tasks.

O	O	0	O	IN	In	NOFUNC	x	O
O	O	1	NP	DT	1801	NOFUNC	x	O
O	O	2	O	,	,	NOFUNC	x	O
O	B-Unknown	3	NP	NNP	Mormon	NOFUNC	x	O
O	O	4	NP	NN	leader	NOFUNC	x	O
Arg1	B-Peop	5	NP	NNP/NNP	Brigham/Young	NOFUNC	x	O
O	O	6	O	VBD	was	NOFUNC	x	O
O	O	7	O	VBN	born	NOFUNC	x	O
O	O	8	O	IN	in	NOFUNC	x	O
Arg2	B-Loc	9	NP	NNP/,/NNP	Whitingham/,/Vt	NOFUNC	x	O
O	O	10	O	.	.	NOFUNC	x	O
5	9	birthplace						

Fig. 1. Dataset sample

III. Dataset Description

This dataset is obtained from the Cognitive Computation Group's Entity and Relation Recognition Corpora[18]. The dataset is given as a single text file that contains entity and relation information about the sentences in the dataset. Each sentence is represented as a block of lines where each line corresponds to a word in the sentence. The information such as the part of speech of the word and the entity are described in the line corresponding to that word. Followed by this block, several lines follow that indicate all the existing relations between entities in the sentence. Meaningful columns in the block of a sentence are:

1) Entity class label (B-Unknown, B-Peop, or B-Loc, which means other_entity, person, location)
2) Element order number
3) Part-of-speech tags
4) Words

The format of a relation descriptor is:

1) First field denotes the location of first entity in the sentence
2) Second field denotes the location of second entity in the sentence
3) Third field denotes the relation between the two entities (e.g. kill or birthplace)

The figure 1 shows all rows and columns in the table for a sample sentence in the dataset.

To understand the dataset distribution, analysis of the dataset is performed. The dataset consists of 928 instances or sentences each containing multiple entities and relations among them. There are a total of 467 relations in the dataset, out of which 268 belong to kill type and 199 belong to birthplace relation shown in figure 2. 764 entities in the dataset belong to type B-Peop (people), 425 entities are of type B-Loc (location) and 515 belong to B-Unknown (unknown) type shown in figure 3. This dataset is used to train, validate, and test the models developed in this work.

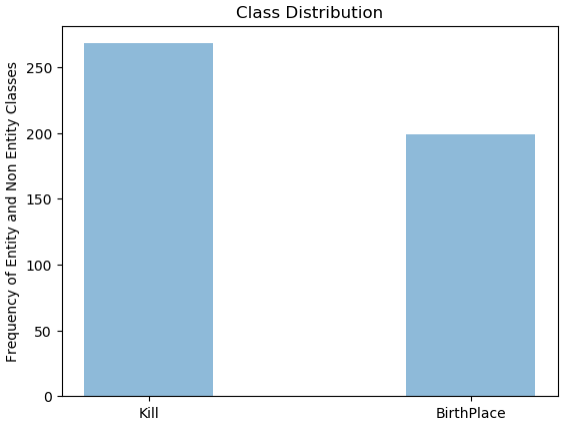

Fig. 2. Relation Distribution

IV. Methodology

In this section, the developed model architectures and their variations are discussed. We used the dataset obtained from cognitive computation group in our experiments. We present three models developed to tackle different problems such as 1) Named Entity Recognition, 2) Relation Extraction and 3) Joint Model.

A. Named Entity Recognition

This is modeled as a sequence labeling problem where each word has to be tagged with a particular entity label. The input to the model is a sentence containing a series of words and the model produces a tag sequence as output that correspond to the entity tags. The devised model consists

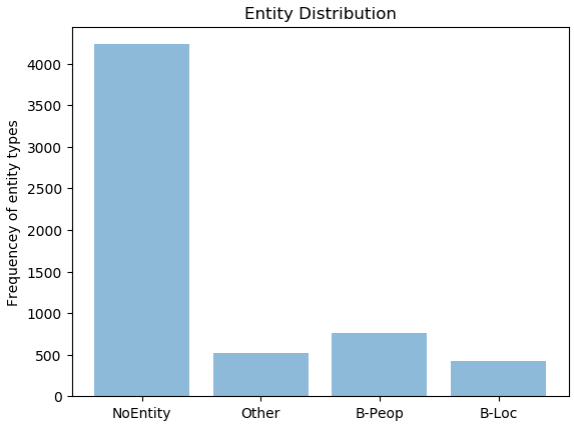

Fig. 3. Entity Distribution

of several layers with the first being the embedding layer followed by a Bidirectional LSTM encoding layer and a conditional random field at the end to identify the most probable tag for that word.

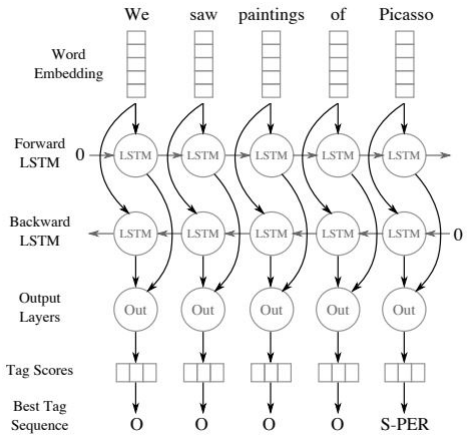

Fig. 4. Neural Model for Named Entity Recognition

Embedding Layer is responsible for mapping each word or token in the sequence to a vector representation. We initialized the embedding weights with Glove pretrained embeddings on Wikipedia dataset comprising of 6B tokens. The loss backpropagates until this layer and weights are updated on each epoch. As the training progresses, the embedding layer weights are adjusted to produce more accurate vector representations for input words. 50 dimensional vector representations are obtained from this layer. These vector representations are propagated forward in the neural network to the upper layers. The experiments prove that initializing the embedding weights with pretrained embeddings results in a better performance.

Next layer in the model is a Bidirectional LSTM, which is a Recurrent Neural Network that capture time dynamics. LSTM is used to model sequence data and to capture long term dependencies. LSTM consists of three multiplicative gates that control the information to be passed on by which it

solves the gradient vanishing problem. Bidirectional LSTM is capable of encoding the information from past context and future as well due to its bidirectional nature. In a unidirectional LSTM, the hidden state takes information only from the past. Bidirectional LSTM considers the sequence in both directions and the final output is a concatenation of both the hidden states. The word representations coming from the embedding layer are fed into the Bidirectional LSTM units that produces a vector for each token in the sequence in which the contextual information is encoded. The model architecture BiLSTM based neural model is shown in figure 4.

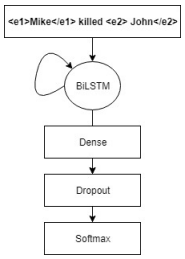

Fig. 5. Neural Model for Relation Extraction

1) Conditional Random Field: To the model described above, we added a conditional random field at the end to enhance the overall performance of the model. Conditional Random Fields are used to decode the labels jointly by considering the neighborhood. It is beneficial to consider the context of labels when predicting the label of the current token. For an input sentence

$$X = (x_1, x_2, ..., x_n)$$

Let us consider A to be the matrix of scores generated from the BiLSTM neural model with softmax. So, given that the input length of the sentence is n, and number of distinct tags is k, the dimensions of A would be $n \times k$, with A_{ij} corresponding to the score of the i^{th} word having the j^{th} tag. For an output sequence

$$y = (y_1, y_2, ..., y_n)$$

We define the linear chain CRF score

$$s(X, y) = \sum_{i=0}^{n} T_{y_i, y_{i+1}} + \sum_{i=1}^{n} A_{i, y_i}$$

where T is a transition matrix with T_{ij} representing the score of transition from i^{th} tag to j^{th}tag with y_0, y_n being the special start and end states. Now instead of local softmax we take a global softmax over the entire output sequence:

$$p(y \mid X) = \frac{e^{s(X,y)}}{\sum_{\tilde{y} \in Y_x} e^{s(X,\tilde{y})}}$$

and during the training we maximize the log-probability of the correct sequence. Softmax layer would take a greedy approach in assigning the labels to tokens by assuming that they are independent of their locality. But for tasks like NER and POS tagging, the independence assumption is not true and a CRF layer improves accuracy compared to Softmax. The output vectors from LSTM units in the previous model are fed to the CRF layer. CRF predicts the most probable entity tag sequence for tokens in the input.

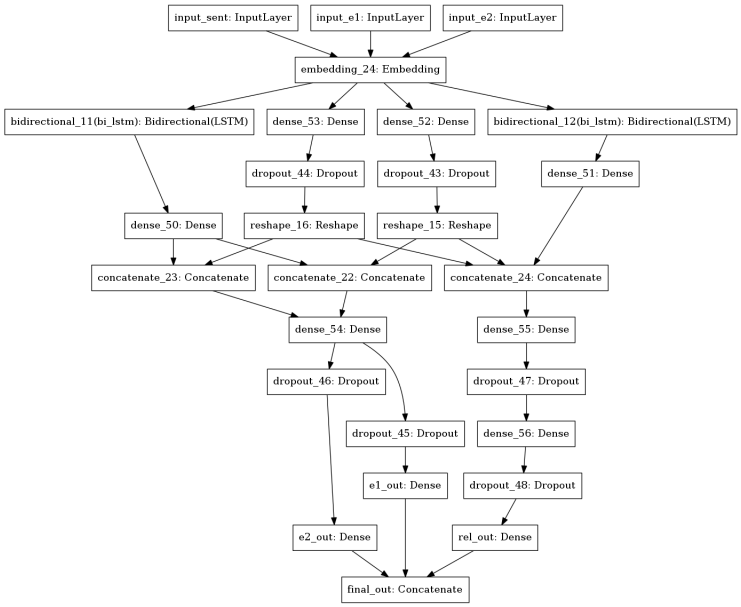

Fig. 6. Neural Model for Joint Entity and Relation Recognition

B. Relation Extraction

For the relation extraction, we assumed that the entity boundaries are known to us. The input to our model is a sentence in which the entity boundaries are marked and only two entities are considered in one instance. If the input sentence has multiple instances, then it is decomposed into multiple combinations and relation between the two entities is encoded appropriately. The output would be one among three categories, no relation, Kill or Birthplace, that precisely identifies how the two entities are related to each other. We used a Bidirectional LSTM to encode the contextual information present in the sentence and extracted the output from the last unit of the network. This output encodes the forward and backward information in the sentence until the last token. This input is fed to a dense feed forward neural network followed by a softmax that predicts the relation category between the entities. Dropout layer has also been added to prevent overfitting. The model architecture is shown in figure 5. The components of this model are similar to the model in NER with the embedding layer followed by a BiLSTM and a softmax at the end.

C. Joint Entity Relation Extraction

Inspired from previous works that tackled the problem jointly rather than with a pipelining approach, we developed a model architecture that is an end-to-end entity relation recognition system. For simplicity, we assumed that there are only two entities in an input sentence that needs to be categorized and one relation between the entities to be recognized. If an input sentence consists of multiple

Task	Models	F1 Score	Precision	Recall
Named Entity Recognition	BiLSTM	0.638180	0.703168	0.592480
	BiLSTM + CRF	0.645264	0.724274	0.592913
Relation Extraction	BiLSTM	0.631062	0.916438	0.560478
Joint Prediction (Entity Scores)	Individual BiLSTMs	0.739614	0.749064	0.738557
	Individual 1d-CNNs	0.746938	0.744517	0.750530
Joint Prediction (Relation Scores)	Individual BiLSTMs	0.772499	0.839335	0.739582
	Individual 1d-CNNs	0.731393	0.776831	0.705122

TABLE I

PERFORMANCE OF DEVELOPED MODELS WITHOUT SEMANTIC LOSS

entities, it is decomposed into combinations where each combination contains a placement of two out of all entities in the sentence. This decomposition enabled us to build an architecture that is relatively simple and solves the problem efficiently. The neural model built is shown in figure 6. The architecture developed would mutually enhance performance of individual models.

The model built is of multi-input type drawing inputs from multiple sources. The input to the model is a sentence marked with entity boundaries along with the two words whose entities and relation between them to be recognized. The output is a triplet that gives the entity tag of the first word, entity tag of the second word and the relation between them. The first layer in the model is an embedding layer that learns the vector representations for the given sequence of input tokens. This layer is shared between entity tagging and relation recognition tasks. The embedding layer generates three separate embeddings for the three inputs to the model. The embedding for the input sentence is fed to a multi-layer 1D-convolutional neural network that creates a sentence representation encoding the contextual information. The embedding resulting from the first input word is concatenated with the vector representation of the sentence from the CNN and the concatenated vector is fed to a fully connected neural network with a softmax loss that outputs the most probable entity tag label for the first word in the input. Similarly, the embedding resulting from the second input word is concatenated with the vector representation of the sentence from the CNN and the concatenated vector is fed to a fully connected neural network with a softmax loss that outputs the most probable entity tag label for the second word in the input. By this point, the model would label the input words with entity tags. For predicting the relation between the entities, we used a separate 1D-convolutional neural network identical to the one before to build a new representation of the given input sentence. This representation is concatenated with the embedding of the first input word and the resulting vector is concatenated with the second input word embedding. The final vector representation encodes the sentence information along with the two words whose relation needs to be categorized. This representation is given as input to a fully connected neural network with dropout followed by a softmax that predicts the most probable relation category between the two entities. The speculation behind using a separate CNN for producing the sentence representations is the CNN weights in the first case are trained and adjusted to provide a representation that captures the entity labels effectively which is different from the requirement in the second case where relation is predicted. The results from the experiments verified our speculations. We have used BiLSTMs instead of CNN in the model architecture and presented the difference in results in the next section.

D. Semantic Loss

(Xu et al., 2018) proposed, semantic loss, a novel methodology for using symbolic knowledge in deep learning. This symbolic knowledge takes form of a Boolean logic constraint. Semantic Loss function enforces logical constraints in a deep learning problem to effectively guide the network train to satisfy the output constraints. The constraint can be as simple as a one hot encoding in output or complex such as ranking or path in graphs. The primary motive behind the semantic loss is to improve the learning performance using the symbolic knowledge. Most end-to-end deep learning systems fail to capture the symbolic knowledge while keeping the model differentiable. The advantage with the semantic loss is that it is differentiable and precisely capture the meaning in the constraints. The semantic loss is treated as an addition to the usual loss functions such as categorical cross entropy. The semantic loss $L(\alpha, p)$ of a prepositional constraint α, is formally defined over the variables $X = X_1, X_2, , X_n$ and the probabilities p over the variables X. Semantic loss can be formulated as follows:

$$L^s(\alpha, p) \propto -log \sum_{x \models \alpha} \prod_{i: x \models X_i} p_i \prod_{i: x \models \tilde{X}_i} (1 - p_i)$$

We have applied semantic loss function to our models in Named Entity Recognition and Joint Entity Relation Extraction. Semantic Loss application to NER falls into one hot encoding case where we enforce a constraint that only one entity label should be predicted as output. This loss function is used along with cross entropy or CRF to effectively guide the model learning.

$$L^s(exactly - one, p) \propto -log \sum_{i=1}^{n} p_i \prod_{j=1, j \neq i}^{n} (1 - p_j)$$

Task	Models	F1 Score	Precision	Recall
Named Entity Recognition	BiLSTM	0.641155	0.725261	0.585539
	BiLSTM + CRF	0.647224	0.683997	0.617281
Joint Prediction (Entity Scores)	BiLSTMs	0.746904	0.759209	0.743120
	1d-CNNs	0.748939	0.755862	0.747127
Joint Prediction (Relation Scores)	BiLSTMs	0.785673	0.795898	0.799343
	1d-CNNs	0.787259	0.806101	0.785976

TABLE II

PERFORMANCE OF NEURAL MODELS WITH SEMANTIC LOSS

In the joint model where the entities and relation between them is recognized jointly, the loss is based on logical constraints. The intuitions behind developing constraints are as follows. Kill relation can only exist between two persons. In other words, the relation kill between two entities implies that the two entities are of type person. Similarly, the birthplace relation can exist only between a person and a location. In other words, the relation birthplace between two entities implies that one of the two entities is a person and the other belongs to type location. The ordering between entities is fixed to avoid relation classification of converse statements based on the actual order. The mathematical notation of semantic loss for birthplace relation is shown below.

$$L^s(birthplace, p) \propto -log(\overline{r_{birthplace}} + r_{birthplace} \times e1_{person} \times e2_{location}) \quad (1)$$

Where $r_{birthplace}$ represents the probability of relation birthplace between the two entities and $e1_{person}$ and $e2_{location}$ represent the probability of the respective entities with first being person and second being location.

Our joint model produces three different outputs at the end, entity type of first word, entity type of second word and the relation between them. These constraints are applied over the probability values resulting from the three penultimate output layers that comprises our semantic loss. Our final loss function is the Semantic loss added with the categorical cross entropy loss.

$$final_{loss} = categorical_{loss} + \lambda(semantic_{loss})$$

The loss is back propagated to all components in the model and respective weights are adjusted. We show the effectiveness of using semantic loss in the results section.

V. RESULTS AND DISCUSSION

We evaluated the models presented in the previous section with the dataset obtained from the cognitive computation group. All of the models are trained with and without semantic loss which enables us to compare the importance of semantic loss for entity relation extraction. We divided the dataset into 80% train and 20% test splits. For the BiLSTM model we built for NER, we achieved an F1 score of 0.638 without using the CRF layer. As speculated, the addition of CRF layer has improved the F1 score to 0.645. The global normalization is beneficial for sequence tagging tasks as concluded by our results. The model built for relation extraction has an F1 score of 0.631. We presented two scores for our joint model, one belongs to the entities and other belongs to relations. We obtained an F1 score of 0.73 which is the average F1 score of prediction for first and second entities in our model. In our experiments, we modified the architecture by replacing BiLSTM with CNNs in our network. This modification turned to be a bit more accurate for entity classification. It resulted in an F1 score of 0.746. In the next row of table-1, the performance scores of the joint model for relation extraction has been presented. In the architecture where BiLSTM is used, we achieved an F1 score of 0.772 and an F1 score of 0.731 when a CNN is used. The results clearly indicate that joint models mutually enhance performance of both entity and relation recognition tasks compared to thier individual counterparts. For the relation extraction, LSTM turned out to be more efficient. In the experiments, overfitting is a concern due to the less number of instances in our training dataset. To alleviate this problem, dropout layers have been embedded into our models. Adding dropout layers was observed to be beneficial during the validation and test phases.

In the table-2, we provided the evaluation metrics of the models after incorporating the semantic loss. This semantic loss is considered as an addition to the typical loss. The value of λ is fixed to be 0.1 after thorough experimentation. For the model built for NER, we achieved an F1 score of 0.641 and an F1 score of 0.647 with CRF layer. We observe that there is a minute performance rise with semantic loss addition for this case. We attribute it to the fact that CRF and Softmax were also trained to a one hot encoding constraint on output which is exactly the constraint encoded in the semantic loss. So, the semantic loss addition has not shown a significant effect in this case. Similarly, the F1 score of the joint model for entity classification with semantic loss is approximately same as the case without semantic loss. We observed an F1 score of 0.746 with BiLSTMs and 0.748 with CNN. The impact of using symbolic knowledge in deep learning and the importance of encoding constraints in a loss function is evident in the relation extraction scores of our joint model. The relation tried to predict is birthplace relation between entities. The semantic loss encodes the constraint that the first entity must be a person and second entity be a location type for this relation to hold. Probabilities of all the cases where this constraint is true is summed up in the semantic

loss equation. We observed an F1 score of 0.78 with both BiLSTM and CNN which is an improvement over previous case.

VI. CONCLUSIONS

In this paper, we have emphasized the importance of capturing symbolic knowledge for a natural language processing application. We have also focused on the effectiveness of joint model that benefits from shared learning of entity tagging and relation recognition rather than treating these two tasks separately. The semantic loss function which encodes logical constraints of the problem is proven to be useful for improving the performance of the deep learning models and suggests its utility to other applications.

ACKNOWLEDGMENT

I would like to thank my project advisers Prof. Kai-Wei Chang and Prof. Guy Van den Broeck for their consistent support.

REFERENCES

[1] Katiyar, A., & Cardie, C. (2017). Going out on a limb: Joint extraction of entity mentions and relations without dependency trees. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (Vol. 1, pp. 917-928).

[2] Xu, J., Zhang, Z., Friedman, T., Liang, Y., & Broeck, G. V. D. (2017). A semantic loss function for deep learning with symbolic knowledge. arXiv preprint arXiv:1711.11157.

[3] Miwa, M., & Bansal, M. (2016). End-to-end relation extraction using lstms on sequences and tree structures. arXiv preprint arXiv:1601.00770.

[4] Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:1603.01354.

[5] Bekoulis, G., Deleu, J., Demeester, T., & Develder, C. (2018). Joint entity recognition and relation extraction as a multi-head selection problem. arXiv preprint arXiv:1804.07847.

[6] Lafferty, J., McCallum, A., & Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

[7] Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin Markov networks. In Advances in neural information processing systems (pp. 25-32).

[8] Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004, July). Support vector machine learning for interdependent and structured output spaces. In Proceedings of the twenty-first international conference on Machine learning (p. 104). ACM.

[9] Zhou, G., & Su, J. (2002, July). Named entity recognition using an HMM-based chunk tagger. In proceedings of the 40th Annual Meeting on Association for Computational Linguistics (pp. 473-480). Association for Computational Linguistics.

[10] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360.

[11] Kambhatla, N. (2004, July). Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In Proceedings of the ACL 2004 on Interactive poster and demonstration sessions (p. 22). Association for Computational Linguistics.

[12] Zelenko, D., Aone, C., & Richardella, A. (2003). Kernel methods for relation extraction. Journal of machine learning research, 3(Feb), 1083-1106..

[13] Santos, C. N. D., Xiang, B., & Zhou, B. (2015). Classifying relations by ranking with convolutional neural networks. arXiv preprint arXiv:1504.06580.

[14] Xu, Y., Mou, L., Li, G., Chen, Y., Peng, H., & Jin, Z. (2015). Classifying relations via long short term memory networks along shortest dependency paths. In Proceedings of the 2015 conference on empirical methods in natural language processing (pp. 1785-1794).

[15] Vu, N. T., Adel, H., Gupta, P., & Schtze, H. (2016). Combining recurrent and convolutional neural networks for relation classification. arXiv preprint arXiv:1605.07333.

[16] Miwa, M., & Sasaki, Y. (2014). Modeling joint entity and relation extraction with table representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1858-1869).

[17] Adel, H., & Schtze, H. (2017). Global normalization of convolutional neural networks for joint entity and relation classification. arXiv preprint arXiv:1707.07719.

[18] https://cogcomp.org/page/resource_view/43.