Knowledge Graph Generation with Deep Active Learning

Abhishek Pradhan*
Research & Development
Taiger Singapore Pte. Ltd
Singapore
abhishek.pradhan@taiger.com

Ketan Kumar Todi*
Research & Development
Taiger Singapore Pte. Ltd
Singapore
ketan.todi@taiger.com

Anbarasan Selvarasu
Research & Development
Taiger Singapore Pte. Ltd
Singapore
anbarasan.selvarasu@taiger.com

Atish Sanyal
Research & Development
Taiger Singapore Pte. Ltd
Singapore
atish.sanyal@taiger.com

***Abstract*—Creating a knowledge graph automatically from raw unstructured text has always been a job of domain expert which takes months to curate and refine. In this paper, we propose a domain-independent semi-automatic knowledge graph learning system that can be trained with less amount of data, to identify entities and relations from a large text corpus. The system performs the following tasks to extract knowledge graph from the text: *(i)* Named Entity Recognition (NER), and *(ii)* Relation Identification (Open Relation Extraction (OpenRE) and Classification). The system uses deep active learning to calculate confidence scores using maximum normalized log-probability on each prediction for both NER, and relation identification. We experimented with both LSTM and transformer based models for NER and relation identification tasks.**

We achieved around 88% F1 score for the NER task on OntoNotes-5.0 English data set with 40% training data set and above 83% F1 score for relation identification on TACRED dataset. The OpenRE and relation classification systems were trained on domain-specific datasets. To the best of our knowledge, we are the first to introduce a knowledge graph generation learning system with deep active learning.

***Index Terms*—Active Learning, Named Entity Recognition, Open Relation Extraction, Knowledge Graph, Open Information Extraction**

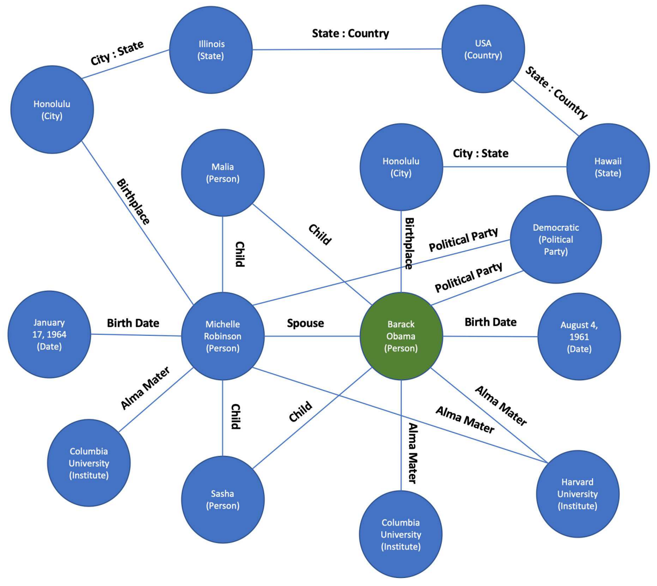

Fig. 1. Sample Knowledge Graph of Barack Obama

I. Introduction

A knowledge graph [1] is defined as the graphical knowledge representations of entities, represented as nodes in the graph, and the relations, represented as edges in the graph, between them. Figure 1 explains the basic structure of a knowledge graph. Each circle in the figure represents a node and a line connecting two circles represent a relation between two entities in the knowledge graph. Each circle has the value of the entity and its type as a property of the node. DBpedia [2], YAGO [3] and Freebase [4] are some of the well known knowledge graphs.

Knowledge Graph (KG) Learning (Ontology Learning, Ontology Extraction, or Knowledge Graph Extraction) is the process of learning to extract entities and relations automatically without any human supervision or semi-automatically with some human supervision. KG learning includes learning of extraction of domain specific terms or entities and the relationship between these entities from a large corpus of unstructured text [5]. Building large knowledge graphs manually takes huge amount of labour and time. Automating the knowledge graph creation with little human supervision not only leads to significant reduction of initial effort of creating the knowledge graph but also helps in enriching the existing knowledge graph by identifying new entities and relations from new text.

* Equal Contribution

Automating knowledge graph extraction involves following two major tasks: *(i)* Named Entity Extraction and *(ii)* Relation Identification.

*a) **Named Entity Recognition**:* Named Entity Recognition is the task of identifying the entities and classifying them into their types. For example, *Barack Obama, the first African American president of United States of America was succeeded by Donald Trump*. Here *Barack Obama* is an entity and its type is *Person*.

A trainable NER model was used to make the system domain-independent and deep active learning was used to reduce the data required for domain-specific entity recognition. The NER model was trained on publicly available CoNLL 2003 NER [6] dataset and then fine-tuned on OntoNotes Release 5.0 [7] with deep active learning. The NER training and fine-tuning was done without any external knowledge source to keep the model domain independent. The NER model can easily be adapted to any kind of unstructured data and any number of NER labels by modifying the final feed-forward layer with new NER labels.

*b) **Relation Identification**:* Open Relation Extraction (OpenRE) [8] is the task of identifying the word or the sequence of words, which describes the relationships between the entity pairs, in a sentence. OpenRE task is very different from relation classification, which aims to classify the relationship between two entities into one of the predefined relationship class. The problem with relation classification is that user needs to know all the types of relationships that can appear in the huge corpus, which is impractical. Therefore, our system learns to identify relations from the text in addition to classifying the relationship in the text. OpenRE also differs from Open Information Extraction (OpenIE) [9] as the latter aims to identify the verb connecting two arguments while the former aims to identify noun and verb relations between entities extracted by NER module. In OpenIE, arguments may or may not be an entity whereas in OpenRE both arguments must be an entity.

Similar to NER module, a trainable OpenRE module was created to keep the module domain independent and easily adaptable to deep active learning. OpenRE task can easily be converted to sequence labelling task [10] where the identified relation is a sequence of words in the sentence. To identify the direction of relationship in the sentence, the entities were classified as either entity 1 or entity 2, where the output triplet was in the form of *Entity 1* $\rightarrow$ *Relation* $\rightarrow$ *Entity 2*.

To improve the recall of identifying relations from the text we also added relation classification. For relation classification we used TACRED dataset [11] which contains the 41 Knowledge Graph Population (KGP) classes.

II. Related Work

Extracting knowledge graph from text involves following two major steps: *(i)* Named Entity Recognition and *(ii)* Relation Identification. End-to-end knowledge graph extraction [12] have few more steps like Entity and Triplet mapping to an existing knowledge graph. Entity and Triplet mapping to an existing knowledge graph is out of scope of this work. This work focuses on creating a system that can learn to identify entities and create relationship links between entities with less amount of data i.e. for those domains where only raw text is available and no existing entities or relations are defined or available publicly.

NER is one of the most important tasks in NLP and is the first step in generating knowledge graph (KG) automatically. Modern NER systems use deep neural networks [13]. Traditionally, Recurrent Neural Networks i.e., LSTM and GRU based NER systems [14] [15] have been widely used in production and academic research. Recently, language model based transformer models [16] [17] have been shown to improve the accuracy further.

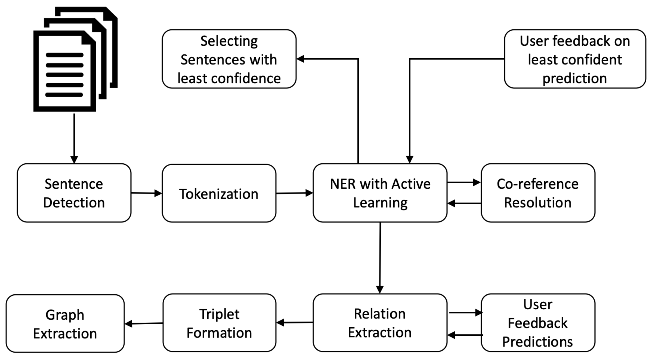

Fig. 2. Overall Architecture

OpenRE is a very important step in building knowledge graph (KG). OpenRE problem is sometimes solved using Open Information Extraction (OpenIE) techniques [18] [12]. The accuracy of extracting open relations using OpenIE is quite low and the accuracy drops further if complex sentences are encountered. Reverb [19] and Supervised Open Information Extraction [20] in AllenNLP framework [9] are some of the widely used Open Information Extraction frameworks.

Unlike OpenIE, OpenRE technique aims to identify the word or sequence of words within the sentence that specifies a relation between two entities. Traditionally, relation classification tasks have been confined to identifying and classifying a relation fact into predefined relationships types [21] [8]. This makes identifying new kinds of relations automatically from raw text difficult.

Traditionally, hand-crafted lexical and syntactic patterns [19] [22] were used to solve OpenRE problems. Although these rule-based approaches were very successful, they do not generalize well on complex and diverse open relations. Neural network based OpenRE [10] have been very successful in improving both the overall accuracy and the domain adaptation.

OpenRE methods are generally divided into two major categories i.e., *(i)* Unsupervised Learning methods and *(ii)* Supervised Learning methods [10]. Unsupervised learning methods extract semantic patterns using linguistic features and cluster them to identify open relations [23] [24] [25]. Identifying linguistic features often require external tools. Supervised learning methods convert the OpenRE problem into sequence tagging problem [20] [26] and can be solved end-to-end.

Creating knowledge graph often requires huge amount of tagged data for both NER and OpenRE. Deep active learning [27] based methods solve this issue by selecting sentences where the system is least confident in predicting the tags and ask user to provide correct tags. Deep active learning based NER architecture [27] has been able reduce the data requirement drastically without affecting much on accuracy.

In this work, we have combined BERT based NER with active learning, co-reference resolution, open relation extraction and relation classification tasks. OpenRE is responsible for extracting noun and verb relationships from the sentences, whereas relation classification involves classifying KGP relations from TACRED dataset [11]. To the best of our knowledge, we are the first to combine OpenRE and relation classification with active learning for knowledge graph population.

III. System Architecture

In this section, the overall architecture of end-to-end knowledge graph extraction is described. The overall architecture of our knowledge graph extraction system is shown in Figure 2. Our knowledge graph extraction learning system has following components: *(i) Named Entity Extraction:* Identifying domain specific named entity from the text, and *(ii) Relation Identification:* Identifying open relations from the text and classifying relations into one of the existing knowledge graph population (KGP) relations. The detailed explanation of each step is provided in next sections.

A. Preprocessing

Spacy [28] library was used for splitting the sentences, tokenization, and Part-of-Speech (POS) tagging. POS tags were later used for training purposes. Sentences in which all the words were tagged as *"O"* (Others) were removed from the training dataset as they served no useful purpose.

B. Active Learning

Active Learning aims to reduce the labelled data required for training. Small batches of sentences were selected in each iteration of Active Learning for further labelling and training. In this work, we used Random Selection and Maximum Normalized Log-Probability (MNLP) [27] method for reducing the number of tagged data required.

*1) **Random Selection**:* In this method, the system randomly identifies a set of sentences from the unlabelled dataset. Using Random Selection for selecting sentences in initial stages of LSTM-CNN model helps model learn quickly.

*2) **Maximum Normalized Log-Probability (MNLP)**:* In this method, we used the normalized log-probability scores for selecting the sentences to be tagged. The log probability score represents the total probability for the entire predicted sequence. Since longer sentences get higher scores due to larger sequence length, the function tends to pick smaller sentences for manual labelling. Therefore, normalizing the probability scores alleviates the above mentioned problem. The sentences are first sorted in ascending order on the basis of the normalized probability score and the sentences with the lowest scores are selected for labelling. The log probability function can be mathematically represented as Equation (1) :

$$PredictionConfidence =$$

$$max_{y_1,...,y_n}(P[y_1,...,y_n|\{x_j\}_{j=1,...,n}]) \Leftrightarrow$$

$$max_{y_1,...,y_n}\prod_{i=1}^{n}P[y_i|y_1,...,y_{i-1},\{x_j\}_{j=1,...,i}] \Leftrightarrow$$

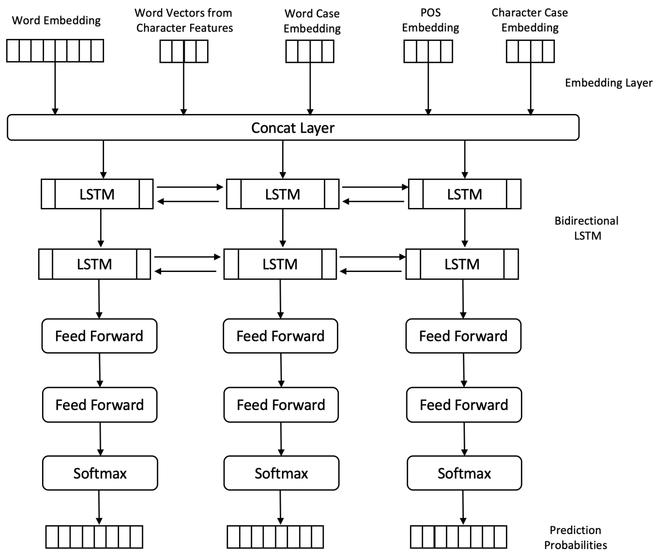

Fig. 3. NER Architecture

$$max_{y_1,...,y_n}\sum_{i=1}^{n}logP[y_i|y_1,...,y_{i-1},\{x_j\}_{j=1,...,i}] \quad (1)$$

The normalized log probability function can be mathematically represented as Equation (2):

$$\frac{1}{n}*max_{y_1,...,y_n}\sum_{i=1}^{n}logP[y_i|y_1,...,y_{i-1},\{x_j\}_{j=1,...,i}] \quad (2)$$

C. Named Entity Recognition

*1) **Bidirectional LSTM + Character CNN based Named Entity Recognition**:* Named entity recognition is a very important and challenging task. NER is one of the most important component of knowledge graph extraction learning system. The architecture diagram of our NER system is shown in Figure 3. We used Glove [29] embedding and character level CNN to encode the characters in the text to get combined embedding having character-level and word-level features. Learning all the features from the training data kept the architecture generic and domain independent.

*a) **Character-Level Word Embedding**:* Character level embedding is extracted by running Character Level Convolutional Neural Network [30]. The architecture diagram of character-level embedding is shown in Figure 4. A word was split into constituent characters and each character's one-hot encoding was used as input to the CNN network. This representation converts text into 2-Dimensional matrix. Convolution *h(y)* is then computed using Equation (3):

$$h(y) = \sum_{x=1}^{k} f(x)\cdot g(y\cdot d - x + c) \quad (3)$$

where *g(x)* is input function, *f(x)* is kernal function, *d* is stride and *c* is an offset constant, is calculated using Equation (4).

$$c = k - d + 1 \quad (4)$$

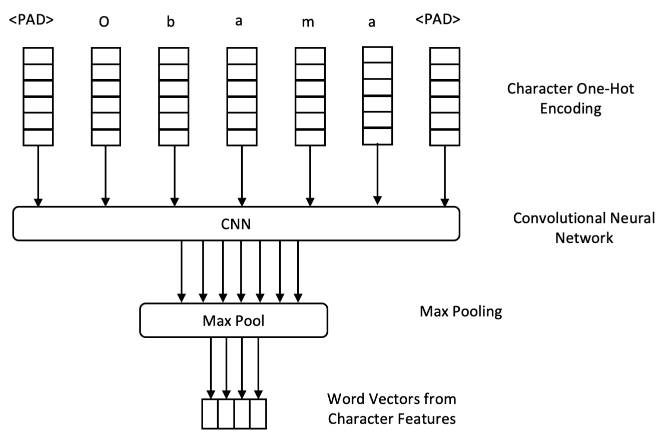
<PAD>
O
b
a
m
a
<PAD>
Character One-Hot Encoding
CNN
Convolutional Neural Network
Max Pool
Max Pooling
Word Vectors from Character Features

Fig. 4. Character Level Embeddings

*b) **Character Case Embedding**:* Character case i.e. upper case, lower case, punctuation, and other were used as features in NER. Architecture similar to character level word embedding was used to extract the embedding from character cases.

*c) **Word Embedding**:* The embedding layer of NER was first initialized with pre-trained Glove [29] embedding and was further fine-tuned during the training.

*d) **Word Case Embedding**:* Word case (i.e. all caps, upper initial, all lower case, mixed caps, number, no info) embedding was created and one-hot encoded. The Embedding layer was initialised with one-hot vectors and was fine-tuned along with NER training.

*e) **Part-of-Speech Embedding**:* Similar to word case embedding, POS embedding was also incorporated so that the neural network can learn the relationship and dependencies between POS tags used in the data.

*f) **Dependency Parser Tags Embedding**:* Similar to POS embedding, dependency parser tags was also incorporated into the model. But there was no significant performance improvement, as a result, it was not incorporated into the final model described above.

*2) **BERT Based Named Entity Recognition**:* Transformer [16] was first introduced for the application of Neural Machine Translation. Since then, transformer based models have achieved state-of-the-art accuracies in many NLP tasks including Named Entity Recognition [17] [31]. BERT [17], a transformer based architecture, was used for our NER module along with Deep Active Learning.

For implementation, we used Hugging Face's [32] transformer package, a PyTorch [33] based transformer framework. We were able to achieve 89% F1 score using *BERT Base Cased* for NER module without making any changes to the architecture.

The main aim of our work was to reduce the data required without reducing the accuracy. We experimented with different training methods for BERT with Active Learning and had some interesting results. For each of the following variations BERT Base Cased Model was used. For every iteration of active learning in all the following approaches, the sentences were selected using the MNLP approach described in Active Learning subsection above.

*a) **Method 1**:* Selected sentences were combined with sentences selected from all previous iterations of active learning. In this case, we did not freeze the BERT language model while training for NER task. In every iteration, the previously trained BERT model was discarded and NER training is done from start.

*b) **Method 2**:* Unlike *Method 1*, in each iteration, the model trained in previous iteration was used for training of further iterations. In this case, we did not freeze the BERT language model while training for NER task. This model performs worse than the *Method 1*. This could be due to the model forgetting the previously learnt language model with each iterations. Also as mentioned in [17] the number of epochs while tuning should be less than or equal to 5, but since in this case the model is being fine-tuned for around 1400 times by the end of the training process, the performance seems to have degraded.

*c) **Method 3**:* In this method, the BERT language model is first fine-tuned on the entire dataset without NER head. Later, the entire model including the BERT language model is fine-tuned in the process similar to the one used in *Method 1*. This model performs similar to the *Method 1*.

*d) **Method 4**:* In this variation, the BERT language model is first fine-tuned on the target dataset without NER head. After, BERT language model is fine-tuned, process similar to *Method 1* was used. Unlike *Method 3*, we froze the BERT language model while training on the NER task. We believe, since the language model was not fine tuned for NER specific task, the performance was not able to cross a certain barrier as shown in figure 5.

D. Coreference Resolution

Coreference Resolution aims to cluster all the co-referring mentions in the text. Many relations in the text appear along with pronouns instead of Named Entities and sometimes the entities are mentioned in the abbreviations or some other forms. Identifying and replacing such mentions with their respective co-referring entity improves the relation extraction accuracy. We used end-to-end coreference resolution [34] model. We added some domain and application specific post processing to the coreference results to standardize our output. The predicted clusters were filtered based on NER Tags.

Words not labelled as *O* (Others) by the NER were not considered as mentions for the named entity and words having tags *O* (Others) were considered as mentions. For example in the paragraph *"Barack Obama, the first African American president of United states of America was succeeded by Donald Trump. The general public place him among the upper tier of American presidents."* Here, the coreference module resolves the mention *the first African American president of United states of America* and *him* to refer to *Barack Obama*. But the first mention is an attribute of *Barack Obama* and also the NER tags are not *O* for all the words in the sequence, so it is not considered as a mention. Resolving such cases would replace the relation information in the sentence with a named entity. This will lead to loss of information from the perspective of constructing a knowledge graph.

E. Relation Extraction and Classification

BERT classifier model was trained to segregate sentences into two categories: *no relation* and others (for relation extraction and KGP Relation classification).

Our OpenRE task can be considered as a sequence labelling task. We experimented two different architectures for the above sequence labelling task, *(i)* inspired by Supervised Open Information Extraction [20], and *(ii)* BERT where it was modelled as an NER task.

For method *(i)*, all possible entity pairs were created from a single sentence and were passed to relation extraction module. The mentions from coreference module of a cluster were replaced with the named entities, the mentions were referring to. Therefore, each sentence in the text can be considered individually for relation extraction task. For method *(ii)*, special token i.e. *_ENT_* at the beginning and end of each entity was added in the sentence to provide entity location information to BERT. MNLP based active learning was also experimented for relation extraction to reduce the data requirements.

For the KGP relation classification, we implemented the model proposed in [35]. The system was experimented with BERT and RoBERTa with MNLP based active learning. Since sentences with relation labelled as *Others* have already been segregated after the classification above, the remaining relation classification dataset had 41 classes.

The results of above experiments are discussed in results section.

IV. Dataset

A. Named Entity Recognition

a) CoNLL-2003: The CoNLL-2003 [6] dataset has 4 types of entities namely PER, ORG, LOC, and MISC. The details regarding the number of tokens and sentences is shown in Table I.

TABLE I
CoNLL Dataset

	# of sentences	# of tokens
Training set	14,041	204,566
Dev Set	3,453	46,666
Test Set	3,250	51,577

b) OntoNotes 5.0: The OntoNotes [7] dataset has 18 types of entities. The details regarding the number of tokens and sentences is shown in Table II. Since the main aim of the project was relation extraction, for which a pair of named entity was required, the OntoNotes dataset was reduced to sentences having more than one type of named entity.

TABLE II
OntoNotes Dataset

	# of sentences	# of tokens
Training set	49,601	1,239,033
Dev Set	5,000	124,148
Test Set	5,089	124,437

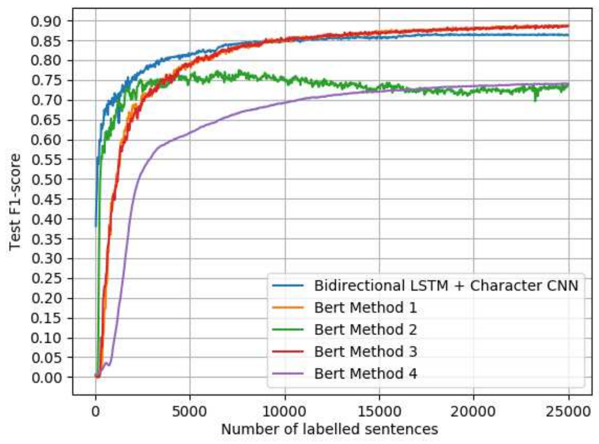

Fig. 5. BERT-NER Test result over full dataset

B. Relation Extraction and Classification

TACRED Dataset [11] was used for OpenRE system. TACRED Dataset was automatically annotated using OpenIE 5.0 [36] [37] [38] [39]. Only triplets from OpenIE 5.0 having a named entity in both the arguments were used to create the dataset for training OpenRE module. The results of OpenIE model was further refined and annotated to identify the word depicting the relation between the two named entities. The details regarding the number of sentences in OpenRE and TACRED dataset is shown in Table III and Table IV respectively.

TABLE III
OpenRE Dataset

	# of sentences
Training set	6528
Dev Set	947
Test Set	839

TABLE IV
TACRED Dataset

	# of sentences
Training set	13,012
Dev Set	5,436
Test Set	3,325

V. Experiments

The two different training methods of active learning were adopted during the training process: *(i)* training the model from scratch in every iteration, and *(ii)* fine tuning the model trained

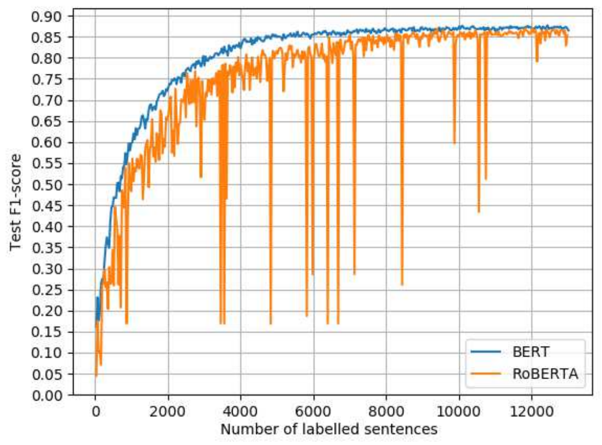

Fig. 6. TACRED classification active learning

TABLE V
NUMBER OF SENTENCES VS F1-SCORE FOR ACTIVE LEARNING BASED NER

Model	**NER 5k**	**NER 10k**	**NER 15k**	**NER 20k**	**NER 25k**
BERT 1	0.796	0.848	0.871	0.879	0.890
BERT 2	0.749	0.748	0.731	0.718	0.734
BERT 3	0.791	0.850	0.867	0.878	0.888
BERT 4	0.616	0.691	0.720	0.734	0.734
LSTM	0.810	0.848	0.856	0.864	0.864

during the previous cycle. The validation set provided in the OntoNotes dataset was modified according to the OntoNotes dataset description in Section IV. For all the active learning based approaches defined above in Section III the following setting were implemented. During every training cycle process for all models , *n (Number of sentences to be annotated in each iteration)* sentences were selected from the training dataset based on one of the active learning approaches described in Section 3. The value of n a trade-off between efficiency and performance. For lower value of n, the speed would be slow as there would be more validation and training cycles, and for large values of n the sentences selected for annotation might have the same kind of uncertainty and may be similar and redundant. In this case, different values of n were experimented, but *n=32* achieved the best results. The selected sentences were labelled and were added to the training dataset. In case of MNLP, the trained model was run on the remaining training dataset and sentences with least confidence scores were selected for next iteration of active learning.

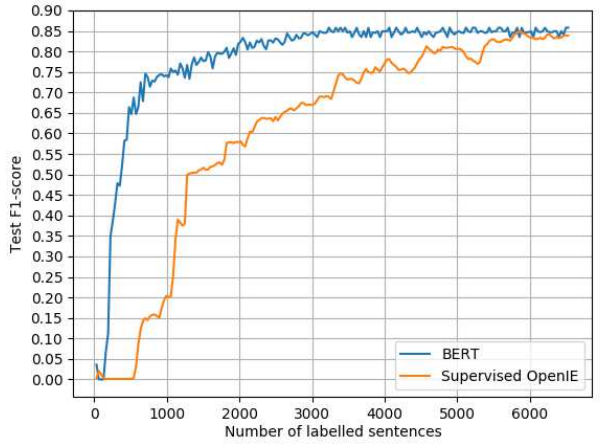

Fig. 7. Relation Extraction active learning

TABLE VI
NER ACTIVE LEARNING RESULTS

Model Name	Test F1-score (maximum accuracy reached)
BiLSTM+CNN	0.866
BERT Method 1	**0.890**
BERT Method 2	0.775
BERT Method 3	0.888
BERT Method 4	0.741

TABLE VII
TACRED CLASSIFICATION

Model Name	Test F1-score (W/o Active learning)	Test F1-score (with Active learning using 40% of data)
BERT	**0.877**	**0.855**
RoBERTa	0.876	0.823

As the model is fine-tuned on the entire dataset in every epoch, having a constant learning rate causes the model to forget the weights it learned previously. The learning rate was divided by 2 in a step wise fashion after every *x* number of iterations, to prevent over-fitting in case of LSTM-CNN NER model, where *x* is a hyperparameter. In our case *x* was set as 300.

$$lr = \begin{cases} 0.001 & \text{if } iterations <= 300 \\ 0.0005 & \text{if } 300 < iterations <= 600 \\ 0.00025 & \text{if } iterations > 600 \end{cases}$$

The batch size for the LSTM-CNN NER model was 8 to have more back propagation steps during the initial training purpose, which helped improve the performance during the initial stages of active learning cycle. A higher batch size also achieved similar final results but in cases where the number of NER labels may be high or the dataset might be complex, achieving better results early in the training process, would help the active learning model to select uncertain sentences in a better manner for further training. Each iteration was run for 10 epochs with early stopping option on the validation loss for 3 consecutive epochs. The number of epochs and early stopping values were decided on the basis of the best performance on validation set. Different experiments with multiple variations of active learning approaches were carried out to maximize the performance. Starting the training process with random

TABLE VIII
RELATION EXTRACTION

Model Name	Test F1-score	Test F1-score with active learning using 40% of test data
BERT	**0.858**	**0.831**
Supervised OpenIE	0.843	0.640

selection and then changing it to MNLP based sentence selection after 200 cycles of training produced better results. The motivation behind this approach is that since the model has very less knowledge about the OntoNotes dataset initially, the sentences selected for labelling from these predictions will have the same effect as following a random selection approach. Therefore, adopting a random selection approach would save the inference time and speed up the training process initially.

In case of BERT based NER (using bert-base-cased), all the hyper-parameters were kept the same throughout the training process. Only MNLP based active learning approach was adopted in BERT due to the inherent language capabality of BERT since it is trained on a huge English corpus. Multiple training methods were carried out for BERT based NER as described in Section III. The training batch size was set as 8, to increase the number of back propagation steps during each cycle of active learning, as the training starts with a small number of labelled sentences, with small addition of new labelled data. The number of epochs during each active learning cycle is kept as 2, consistent with the suggestion for the number of epochs for fine tuning as suggested in BERT paper [17]. The learning rate was also unchanged and was set as $5e-5$ in this experiment.

For the BERT and RoBERTa based relation classification and BERT based extraction tasks defined above, all the hyper-parameters of BERT (bert-base-cased) and RoBERTa (roberta-base) were kept unchanged. The number of epochs, learning rate and the batch size was same as above. For Supervised Open Information Extraction experiment batch size was set as 16, learning rate was set as 0.001 and number of epochs as 2, to prevent over-fitting due to the small size of available dataset.

VI. RESULTS

The LSTM-CNN based NER model described in Section III was able to achieve 88.71% F1 score when trained tested on CoNLL dataset. It was then fine-tuned on the OntoNotes dataset using the active learning approach described in Section III, reached 86% F1 score on the OntoNotes dataset by using 20k (approximately 40%) sentences from the OntoNotes train dataset.

BERT NER model had 89.3% F1 score on the entire dataset. We achieved varying results from the different BERT based NER variations described in Section III. The result comparison is shown in Figure 5 and Table VI.

Figure 5 shows a comparison between the number of sentences used for training and the test F1 score for different NER Models. Table V shows how the test F1 score changed for different active learning based NER models with the increase in number of training sentences. BERT Method 1 and BERT Method 3 out performed all other models. This experiment shows that training method used to train BERT with active learning affects the final accuracy and number of examples required. Freezing BERT does not help in improving the accuracy. Method 2 starts learning quickly but does not improve after certain sentences whereas method 1 and 3 learn slowly but achieves the highest F1 score.

Neural Open Relation Extraction model on OpenRE dataset prepared from TACRED as described in Section IV, was able to achieve F1 score of 85%. With active learning, the data requirement reduced by approximately 60% to achieve similar results without active learning. The result comparison is shown in Table VIII. Supervised OpenIE model requires more sentences than BERT to attain similar performance. One of the main reason for this can be the language model learnt by the base BERT model as compared to the use of GLOVE embeddings in Supervised OpenIE.

Figure 6 shows the comparison between the number of sentences used for training and the test F1 score for BERT and RoBERTA for relation classification. RoBERTA seems unstable with active learning whereas BERT was quite stable while training with active learning. We believe although BERT and RoBERTA have same number of training parameters, RoBERTA requires more examples than BERT to achieve similar accuracy. The stability improves after training RoBERTA for a long time. With active learning, the data requirement reduced by approximately 60% to achieve similar results without active learning.

VII. CONCLUSION AND FUTURE WORK

Creating knowledge graph using active learning reduces the effort required to create knowledge graph from text. More experimentation can be done on coreference resolution and OpenRE. In future work we would like to use ensemble methods to improve OpenRE results.

Sentence simplification before coreference resolution can be used to reduce the complexity of sentences and improve the performance of the relation extraction modules.

As all models are trained on top of BERT, multi-task learning can be experimented to combine and train all heads together.

REFERENCES

[1] H. Paulheim, “Knowledge graph refinement: A survey of approaches and evaluation methods,” *Semantic web*, vol. 8, no. 3, pp. 489–508, 2017.

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

[3] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: A core of semantic knowledge,” in *Proceedings of the 16th International Conference on World Wide Web*, ser. WWW ’07. New York, NY, USA: ACM, 2007, pp. 697–706. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242667

[4] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: A collaboratively created graph database for structuring human knowledge,” in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’08. New York, NY, USA: ACM, 2008, pp. 1247–1250. [Online]. Available: http://doi.acm.org/10.1145/1376616.1376746

[5] K. Ahmad and L. Gillam, “Automatic ontology extraction from unstructured texts,” in *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE*, R. Meersman and Z. Tari, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 1330–1346.

[6] E. F. Sang and F. De Meulder, “Introduction to the conll-2003 shared task: Language-independent named entity recognition,” *arXiv preprint cs/0306050*, 2003.

[7] R. Weischedel, S. Pradhan, L. Ramshaw, J. Kaufman, M. Franchini, M. El-Bachouti, N. Xue, M. Palmer, J. D. Hwang, C. Bonial *et al.*, “Ontonotes release 5.0,” 2012.

[8] D. Yu, L. Huang, and H. Ji, “Open relation extraction and grounding,” in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 854–864. [Online]. Available: https://www.aclweb.org/anthology/I17-1086

[9] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. Peters, M. Schmitz, and L. S. Zettlemoyer, “Allennlp: A deep semantic natural language processing platform,” 2017.

[10] R. Wu, Y. Yao, X. Han, R. Xie, Z. Liu, F. Lin, L. Lin, and M. Sun, “Open relation extraction: Relational knowledge transfer from supervised data to unsupervised data,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 219–228.

[11] Y. Zhang, V. Zhong, D. Chen, G. Angeli, and C. D. Manning, “Position-aware attention and supervised data improve slot filling,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, 2017, pp. 35–45. [Online]. Available: https://nlp.stanford.edu/pubs/zhang2017tacred.pdf

[12] N. Kertkeidkachorn and R. Ichise, “T2kg: An end-to-end system for creating knowledge graph from unstructured text,” in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[13] V. Yadav and S. Bethard, “A survey on recent advances in named entity recognition from deep learning models,” *arXiv preprint arXiv:1910.11470*, 2019.

[14] J. P. Chiu and E. Nichols, “Named entity recognition with bidirectional lstm-cnns,” *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

[15] S. Misawa, M. Taniguchi, Y. Miura, and T. Ohkuma, “Character-based bidirectional lstm-crf with words and characters for japanese named entity recognition,” in *Proceedings of the First Workshop on Subword and Character Level Models in NLP*, 2017, pp. 97–102.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[18] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, “Open information extraction from the web.” in *Ijcai*, vol. 7, 2007, pp. 2670–2676.

[19] A. Fader, S. Soderland, and O. Etzioni, “Identifying relations for open information extraction,” in *Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2011, pp. 1535–1545.

[20] G. Stanovsky, J. Michael, L. Zettlemoyer, and I. Dagan, “Supervised open information extraction,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 885–895.

[21] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, “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*. Suntec, Singapore: Association for Computational Linguistics, Aug. 2009, pp. 1003–1011. [Online]. Available: https://www.aclweb.org/anthology/P09-1113

[22] Y. Xu, M.-Y. Kim, K. Quinn, R. Goebel, and D. Barbosa, “Open information extraction with tree kernels,” in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013, pp. 868–877.

[23] D. Lin and P. Pantel, “Dirt@ sbt@ discovery of inference rules from text,” in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 323–328.

[24] D. Marcheggiani and I. Titov, “Discrete-state variational autoencoders for joint discovery and factorization of relations,” *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 231–244, 2016.

[25] H. Elsahar, E. Demidova, S. Gottschalk, C. Gravier, and F. Laforest, “Unsupervised open relation extraction,” in *European Semantic Web Conference*. Springer, 2017, pp. 12–16.

[26] S. Zheng, F. Wang, H. Bao, Y. Hao, P. Zhou, and B. Xu, “Joint extraction of entities and relations based on a novel tagging scheme,” *arXiv preprint arXiv:1706.05075*, 2017.

[27] Y. Shen, H. Yun, Z. C. Lipton, Y. Kronrod, and A. Anandkumar, “Deep active learning for named entity recognition,” 2017.

[28] M. Honnibal and I. Montani, “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing,” 2017, to appear.

[29] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[30] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” 2015.

[31] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[32] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, “Huggingface’s transformers: State-of-the-art natural language processing,” *ArXiv*, vol. abs/1910.03771, 2019.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[34] K. Lee, L. He, M. Lewis, and L. Zettlemoyer, “End-to-end neural coreference resolution,” *arXiv preprint arXiv:1707.07045*, 2017.

[35] S. Wu and Y. He, “Enriching pre-trained language model with entity information for relation classification,” in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2361–2364.

[36] S. Saha *et al.*, “Open information extraction from conjunctive sentences,” in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 2288–2299.

[37] S. Saha, H. Pal *et al.*, “Bootstrapping for numerical open ie,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2017, pp. 317–323.

[38] H. Pal *et al.*, “Demonyms and compound relational nouns in nominal open ie,” in *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, 2016, pp. 35–39.

[39] J. Christensen, S. Soderland, O. Etzioni *et al.*, “An analysis of open information extraction based on semantic role labeling,” in *Proceedings of the sixth international conference on Knowledge capture*. ACM, 2011, pp. 113–120.