Relation Extraction in SemEval 2010 Task 8

Mohammed Mehdi Patel, Antas Singh, Nilakhi Das, Ayush Garg
Department of Computer Science
The University of Manchester, UK
mohammedmehdi.patel@postgrad.manchester.ac.uk
antas.singh@postgrad.manchester.ac.uk
nilakhi.das@postgrad.manchester.ac.uk
ayush.garg@postgrad.manchester.ac.uk

***Abstract*—In this paper we describe two approaches for extracting relations on the SemEval 2010 Task 8 dataset. Relation Extraction(RE) is a task that is concerned with identifying the type of relationship, if any, that holds between two given named entities, based on the sentence in which they were mentioned. We employ two approaches, one being a traditional machine learning based approach in which we use an SVM model and the other being a deep learning based approach that uses a Bidirectional LSTM neural network. We used Word Embeddings based on the dataset's corpus of words and generated features using NLP techniques that were used in the models. Through systematic experimentation, we analyze the performance of the two models on the said dataset and compare their effectiveness using metrics like accuracy, precision, recall and f1-score.**

I. Introduction

Relation extraction is the task of identifying entities and their semantic relationships from texts. Its a crucial task in natural language processing, involves identifying and classifying relationships between entities in unstructured text. Over the years, researchers have explored a plethora of techniques, ranging from traditional rule-based methods to advanced machine learning approaches, with a growing emphasis on the application of deep neural networks. This paper delves into the realm of relation extraction, aiming to contribute novel insights, methodologies, and advancements to enhance the accuracy and efficiency of relation extraction systems, especially in the context of specialized domains.

SemEval 2010 Task 8 is a multi-way classification of mutually exclusive semantic relations between pairs of nominals. The label for each example is chosen from the complete set of ten relations- Cause-Effecf(CE), Instrument-Agency(IA), Product-Producer(PP), Content-Container(CC), Entity-Origin (EO), Entity-Destination (ED), Component-Whole (CW), Member-Collection (MC), Communication-Topic (CT) and Others. The descriptions of this relational entites are mentioned in official paper of the dataset[2]. The distribution of the data grouped by the type of relation is shown in Table 1.

In this paper, we emphasise on the task of relation extraction on the SemEval 2010 Task 8 dataset using 2 methods- A traditional machine learning approach of Support Vector Machine (SVM) and another Deep Learning approach of Long Short Term Memory (LSTM). We used a SVM model with RBF kernel with optimal hyperparameters values. We used a Bi-LSTM based model with custom attention Layer.

In the realm of relation extraction on the SemEval 2010 dataset, the comparison between Support Vector Machines (SVM) and Long Short-Term Memory networks (LSTM) serves as a critical exploration of traditional and deep learning approaches. Support Vector Machines, a classical machine learning technique, have been widely employed for relation extraction due to their ability to handle complex feature spaces. On the other hand, Long Short-Term Memory networks, a type of recurrent neural network, excel in capturing sequential dependencies and have shown success in various natural language processing tasks. The comparison involves assessing the performance of SVM and LSTM in terms of precision, recall, and F1-score on relation extraction tasks within the SemEval Dataset. While SVM may exhibit robustness in handling structured feature representations, LSTM's strength lies in capturing contextual information and patterns within sequential data.

Finally the remaining of the paper is structured as follows. Section 2 reviews the related work. Section 3 describes our methodology of the two approaches along with preprocessing techniques to format data and NLP techniques we used to generate features. Section 4 reports on the results comparing the accuracy and the performance of the approaches on the test set. Section 5 concludes the paper.

Relation Name	Frequency
Cause-Effect	1003
Component-Whole	941
Entity-Destination	845
Entity-Origin	716
Product-Producer	717
Member-Collection	690
Message-Topic	634
Content-Container	540
Instrument-Agency	504
Other	1410

TABLE I: The number of examples available for each relation

II. Related Work

Early work by Gumwon Hong in relation extraction using Support Vector Machine presents a supervised approach to detect and classify relations within the Automatic Content

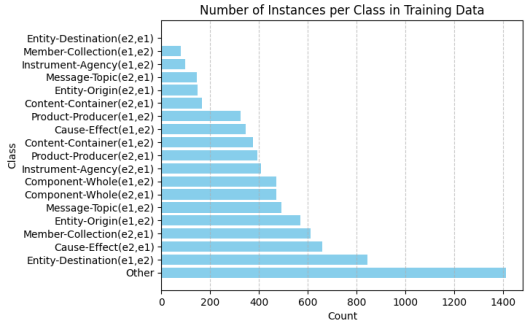

Fig. 1: No. of instances of each class in training data

Extraction (ACE) corpus[3]. The paper discusses the importance of various features, including lexical tokens, syntactic structures, semantic entity types, and particularly the distance between entities to enhance performance. The system is evaluated in terms of recall, precision, and F1-score. In our work, we do not perform relation detection but relation classification. We also use an SVM model with RBF kernel but we find out the optimal values for hyperparameters using grid search. Unlike distance features used by Gumwon Hong, we use word embeddings generated by Google's Word2Vec[1] model for generating features.

Bryan Rink and Sanda Harabagiu in their approach to classifying semantic relations using SVM classifiers, focus on features capturing context, semantic role affiliation, and possible pre-existing relations between nominals[7]. Our research employs a distinct set of features including POS tagging and word2vec embeddings, alongside tf-idf features, for semantic relation classification on the same dataset. Furthermore, we utilize an RBF kernel for the SVM model, where optimal hyperparameters were determined through extensive grid search.

In the context of our study, the research conducted by Lee et al.[4] which employs a Bidirectional LSTM network integrated with entity-aware attention mechanism for semantic relation classification, serves as a critical reference point. Their methodology underscores the importance of focusing on entities within sentences to enhance classification performance. Our work extends this approach by applying a Bidirectional LSTM model with a slightly different architecture and a custom attention layer on the same dataset. We further differentiate our methodology through the incorporation of different features including the utilization of relative positioning with normalization in vector form and Part-of-Speech (POS) tagging.

III. Methodology

Before training our models, we preprocessed the data so that they could be used for generating features for the models. We then train the models with different features generated using NLP techniques. The whole process is described below.

A. Preparing pandas dataframe

The training and testing data files are given in text format. Records in the training and testing files are separated with a new line character. The first line in each record starts with a numerical identifier followed by the sentence in double quotation marks. Two entities in each of the sentences are marked with tags opening tags `</e1>` and `</e2>` and closing tags `</e1>` and `</e2>` respectively. The numbering simply reflects the order of the mentions in the sentence. Below this line, there are two lines mentioning the relation between the two entities and a comment specifying why the annotators chose the relation. An example is shown below:

15 "They saw that the </e1>equipment</e1> was put inside rollout </e2>drawers</e2>, which looked aesthetically more pleasing and tidy." Content-Container(e1,e2) Comment: the drawer contains the equipment, typical example of Content-Container; no movement

The training and testing files in txt format were converted to pandas dataframe using a preprocessing code written in Python. Conversion to pandas dataframe helped understand the number of instances available for each relation type and in further processing of the data, particularly in applying NLP techniques.

B. Numbering the types of relations

There are 9 possible relations that can exist between two entities in a given sentence. Since this is a multi-way classification task, each of the 9 relations can either go from e1 to e2 or from e2 to e1. If none of the relations hold between the two entities, then they are given the label "Other". This makes a total of 19 classes, two for each of the 9 relations and the class "Other". Figure 1 shows the distibution of the training data grouped by the class names. The relations in the pandas dataframe were converted to numbers from 0 to 18. These numbers were then used as labels for the models.

C. Generating Features for SVM Model

Part-of-Speech(POS) Tagging of entities: We used the SpaCy[2] library to tokenize each sentence into individual tokens and find out the POS tags of e1 and e2. We use SpaCy's tokenizer for this purpose. With this tokenizer we find out the POS tags of e1 and e2 and add them as a feature in the pandas dataframe. The feature is in the form of a tuple of 2 elements where the first and second elements contain the POS tag ids of e1 and e2.

Creating Word2Vec Word Embeddings: We built a copus of around 25000 words present in the SemEval 2010 Task 8 dataset. We did this by creating a function in our code and applying it to each sentence in the dataset. We then trained Google's Word2Vec model on this corpus to generate 300-dimensional word embeddings that were used to create features for our models. The word embeddings generated by

[1]https://www.tensorflow.org/text/tutorials/word2vec

[2]https://spacy.io/

the Word2Vec model are 300-dimensional vectors storing contextual information in them. We use these word embeddings to create (1) a vector representing the sentence (2) a vector representing entity e1 (3) a vector representing entity e2.

TF-IDF Features: To highlight the importance of certain words in each relation type, we used TF-IDF vectorizer. TF-IDF vectorizes/scores a word by multiplying the word's Term Frequency (TF) with the Inverse Document Frequency (IDF). Term Frequency(TF) of a term or word is the number of times the term appears in a document compared to the total number of words in the document. Inverse Document Frequency(IDF) of a term reflects the proportion of documents in the corpus that contain the term. The TF-IDF of a term is calculated by multiplying TF and IDF scores. We limit the number of features generated by our TF-IDF vectorizer to 200 for which we want to calculate the TF-IDF scores. We set this parameter value based on multiple experiments. Any value above 200 led to either insignificant improvement or deterioration in the SVM model's performance.

D. Training SVM Model

Support Vector Machines have been proven to be effective in a lot of text classification tasks[8]. They are also able to handle high-dimensional data well[1]. Once the aforementioned features are generated, we train a SVM model. We combine all the features and convert them into a numpy array of 1102 dimensions. We then feed these features to the model employing an rbf kernel and setting the C and gamma hyperparameters to 100 and 0.001 respectively. We find these values to be optimal after conducting a grid search using 4 values each for C and gamma. We select the values for which we get the highest f1-score. Generally, the values of these hyperparameters can have a great impact on a SVM model's performance[5].

E. Features for Bi-LSTM Model

For the Bi-LSTM model, we are using four feature matrices. For the first matrix we use the Keras Tokenizer[3], we take a sentence and transform it into sequences of integers using the fitted tokenizer [6]. Each word in the sentences is replaced with its corresponding integer index from the tokenizer's vocabulary.

For the second and third feature matrices, we leverage the relative distances between the tagged entities (`<e1>` and `<e2>`) and the remaining words within the sentence. These matrices are created separately for each entity (`<e1>` and `<e2>`). For each entity, the words within the sentence are indexed based on their proximity to the respective entity, with the entity itself positioned at the center. To enhance the training effectiveness of our model, we normalize the obtained indexes in both cases, ensuring consistency and facilitating optimal utilization of positional information during model training. This feature is valuable for relation extraction because it encodes spatial relationships between entities and other words in the sentence, providing contextual information crucial for understanding the semantic associations between entities.

For the fourth feature matrix, we employ Part-of-Speech (POS) tagging. Utilizing the SpaCy library, we extract the POS tag for each word in the sentence, assigning a numerical value to represent each tag. This process is applied to every sentence in the dataset, enabling the creation of a feature matrix where each word is represented by its corresponding numerical POS tag. This feature is beneficial for relation extraction as it provides syntactic information about the words in the sentence. By incorporating POS tagging, the model can leverage syntactic cues to better understand the structure and semantics of sentences.

For all four feature matrices, padding is applied to standardize the input size for model compatibility. The padding ensures uniformity in the input dimensions across samples, enabling consistent processing within the model architecture.

F. Architecture of the BiLSTM based model

The model we made has many layers to it, Four input layers are defined one for each feature as discussed before. An embedding matrix is created from SpaCy word vectors, and an embedding layer is defined using this matrix for the word tokens. Adding word embeddings derived from word vectors is helpful because it allows the model to represent words in a continuous vector space where words with similar meanings are closer together.

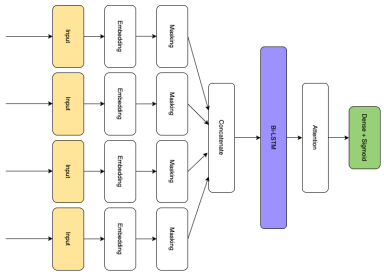

Fig. 2: Bi-LSTM based Model Architecture

Two additional embedding layers are defined for the relative positions of entities, while another embedding layer is defined for the part of speech feature matrix. A masking layer is applied to handle padding in the input sequences. As we have padded our data with '0' we added a mask for '0'.It allows the model to ignore padded values during computation, particularly in sequences of varying lengths.

After masking, the preprocessed input is passed through a Bidirectional Long Short-Term Memory (BiLSTM) layer with 256 units, dropout rate of 0.2, and recurrent dropout rate of 0.3. Bi-LSTM effectively captures long-range dependencies and context from both past and future states of a sequence, enhancing its ability to understand relationships between words in a sentence. Its bidirectional nature allows it to incorporate information from preceding and succeeding words, while LSTM units mitigate the vanishing gradient problem. This enhances the model's understanding of semantic relationships within the input data, crucial for relation extraction tasks.

[3]https://bit.ly/49DJUDQ

An attention layer is added after the Bi-LSTM layer. The attention layer in our model helps by dynamically weighting the importance of each word in the input sequences based on its relevance to relation extraction. It calculates attention scores to identify informative words, applies softmax to obtain attention weights, and then multiplies these weights with the input sequences to generate context vectors. By focusing on relevant parts of the input, the attention mechanism enables the model to capture crucial semantic relationships between words and entities, enhancing its ability to extract meaningful relations accurately. This selective attention improves the model's performance by emphasizing relevant patterns in the input data. A dense output layer with softmax activation is defined to produce the final predictions. The number of units in this layer is 19, which corresponds to the number of classes in the classification task. The model architecture is shown in Figure 2.

G. Training Bi-LSTM based Model

The model is compiled using the Adam optimizer with a categorical cross-entropy loss function and accuracy as the evaluation metric. It is then trained on a training dataset consisting of 8000 data points, utilizing four input features. The training process lasts for 10 epochs with a batch size of 64, and 1% of the training data is used for validation to monitor the model's performance during training. This setup enables the model to learn from the input data and optimize its parameters to minimize the loss function and improve classification accuracy over the training epochs.

IV. Results and Evaluation

After describing the two models used for relation extraction, we proceed to illustrate the dataset utilized and showcase the results for both SVM and Bi-LSTM. The train set consisted of 8000 examples whereas the test set consisted of 2,717 examples. We evaluate both the models on the test set.

The weighted accuracy of Bi-LSTM model turned out to be higher than the SVM model. However, accuracy alone does not truly reflect the performance of our models. Hence, we present our result in terms of recall, precision and F1-Score.

Both the models have zero F1-score for relation Entity-Destination(e2,e1). We believe this is because of the presence of only 1 instance of this relation both in the training and testing data. In most of the relations for which we have enough instances, the models perform better as compared to other relations. However, we observe that despite the lack of data in certain relations, the model is able to perform almost as well as others though there are few exceptions. One such example is class Entity-Origin(e2,e1). It was quite surprising to find that despite having an ample number of instances for class 0(Other) both the models have a low F1-score compared to other classes having less instances. This can be attributed to the lack of training and testing data to help the model identify whether a relation between the two entities exists or not.

Comparing the two models with each other, our Bi-LSTM model outperforms the SVM model consistently in all relations except Entity-Destination(e2,e1). F1 scores of both the models for each class are shown in table 2. For the SVM model, recall ranges from 18% to 82% and precision ranges from 33% to 84%. The recall and precision ranges for the Bi-LSTM model are 39%-88% and 36%-85% respectively. We believe that the SVM model is not able to use the features to generalize as well as the Bi-LSTM. Table 3 shows the weighted precision and recall of both models.

In the presence of class imbalance, to emphasize overall performance across all classes, we use micro F1 score metric. We find that the micro F1 score of the SVM model is 59.2% and that of the Bi-LSTM is 66.1%.

Relation	SVM F1-Score	Bi-LSTM F1 Score
Other	0.39	0.37
Message-Topic(e1,e2)	0.65	0.66
Message-Topic(e2,e1)	0.53	0.59
Product-Producer(e1,e2)	0.51	0.57
Product-Producer(e2,e1)	0.37	0.49
Instrument-Agency(e1,e2)	0.27	0.52
Instrument-Agency(e2,e1)	0.46	0.64
Entity-Destination(e1,e2)	0.78	0.83
Entity-Destination(e2,e1)	0.00	0.00
Cause-Effect(e1,e2)	0.78	0.82
Cause-Effect(e2,e1)	0.81	0.81
Component-Whole(e1,e2)	0.58	0.70
Component-Whole(e2,e1)	0.42	0.59
Entity-Origin(e1,e2)	0.67	0.74
Entity-Origin(e2,e1)	0.70	0.72
Member-Collection(e1,e2)	0.35	0.56
Member-Collection(e2,e1)	0.66	0.80
Content-Container(e1,e2)	0.77	0.79
Content-Container(e2,e1)	0.68	0.75

TABLE II: SVM and Bi-LSTM F1 scores for each class

Model	P	R
SVM	0.61	0.59
Bi-LSTM	0.66	0.66

TABLE III: Weighted Precision and Recall of SVM and Bi-LSTM. P: weighted precision. R: weighted recall

V. Conclusion and Future Work

We performed relation extraction on the SemEval 2010 Task 8 dataset where we used 2 approaches- A traditional machine learning method where Support Vector Machine was used and a Deep Learning based approach where a Bi-LSTM Neural Network was used. Out of the 2 methods, the Bi-LSTM generally performed better than the SVM model in terms of all metrics.

In future work, we plan to improve the models by using lexical features such as word n-grams and distance features such as shortest dependency path between the entities. We also plan to explore Language Models that use the Transformer Architecture(eg: BERT) since they have recently been proved to perform well on a number of text classification tasks. We can fine tune them and test its effectiveness on the same dataset.

REFERENCES

[1] Bissan Ghaddar and Joe Naoum-Sawaya. "High dimensional data classification and feature selection using support vector machines". In: vol. 265. 3. Elsevier, 2018, pp. 993–1004.

[2] Iris Hendrickx et al. "SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals". In: *Proceedings of the 5th International Workshop on Semantic Evaluation*. Ed. by Katrin Erk and Carlo Strapparava. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 33–38. URL: https://aclanthology.org/S10-1006.

[3] Gumwon Hong. "Relation extraction using support vector machine". In: *Natural Language Processing–IJCNLP 2005: Second International Joint Conference, Jeju Island, Korea, October 11-13, 2005. Proceedings 2*. Springer. 2005, pp. 366–377.

[4] Joohong Lee, Sangwoo Seo, and Yong Suk Choi. "Semantic relation classification via bidirectional lstm networks with entity-aware attention using latent entity typing". In: vol. 11. 6. MDPI, 2019, p. 785.

[5] Zhijie Liu et al. "Study on SVM compared with the other text classification methods". In: *2010 Second international workshop on education technology and computer science*. Vol. 1. IEEE. 2010, pp. 219–222.

[6] South Pigalle. "Simple Relation Extraction with a Bi-LSTM Model - Part 1". In: *Medium* (2019). URL: https://medium.com/southpigalle/simple-relation-extraction-with-a-bi-lstm-model-part-1-682b670d5e11.

[7] Bryan Rink and Sanda Harabagiu. "Utd: Classifying semantic relations by combining lexical and semantic resources". In: *Proceedings of the 5th international workshop on semantic evaluation*. 2010, pp. 256–259.

[8] Waleed Zaghloul, Sang M Lee, and Silvana Trimi. "Text classification: neural networks vs support vector machines". In: vol. 109. 5. Emerald Group Publishing Limited, 2009, pp. 708–717.