UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

Relation Classification using LSTM and BERT Models and leveraging Knowledge Graph

Shojaat Joodi Bigdilo *   and Prof. Alfio Ferrara **

*** Author: shojaat.joodibigdilo@studenti.unimi.it**

**** Advisor: Alfio.Ferrara@unimi.it**

17 July 2024

Abstract

In recent years, the integration of large language models (LLMs) into natural language processing (NLP) tasks has significantly advanced knowledge representation and reasoning. This project investigates the application of Long Short-Term Memory (LSTM) networks with a bi-directional architecture and the Bidirectional Encoder Representations from Transformers (BERT) model, utilizing different hyperparameters for relation classification. Additionally, Knowledge Graph (KG) techniques are employed to represent the relation between subjects and objects in the provided texts. The objective is to enhance the efficiency and accuracy of the relation classification task by leveraging the strengths of these models.

Keyword: Natural Language Processing, Relation Classification, LSTM, BERT, Knowledge Graph

1. Introduction

The task of relation classification involves predicting semantic relations between pairs of nominals. Given a text sequence (usually a sentence) s and a pair of nominals e1 and e2, the objective is to identify the relation between e1 and e2 (Hendrickx et al., 2010). This important NLP task is often used as an intermediate step in various NLP applications. For example, consider the sentence: "The [kitchen]e1 is the last renovated part of the [house]e2." Here, the relation between "kitchen" and "house" is classified as Component-Whole. (Suchanek and Yifan, 2019)

Traditional approaches to relation classification often rely on handcrafted features and domain-specific rules, which are limited in their ability to capture the rich contextual information present in natural language.

Recently, transformer-based models like BERT (Bidirectional Encoder Representations from Transformers) have emerged as powerful tools for NLP tasks due to their ability to capture contextual information and learn representations directly from raw text. (Suchanek and Yifan, 2019). In this project, we explore the application of both BERT and Long Short-Term Memory (LSTM) networks with a bi-directional architecture for relation Classification. We aim to leverage BERT's contextual embeddings and the sequential modeling capabilities of LSTM to accurately classify relationships between entities. Additionally, we employ the 'networkx' library to construct and display Knowledge Graphs, representing the relationships between entities (subjects and objects) using provided predicates that indicate their relations. This approach allows for a more comprehensive understanding of the relationships within the text.

Our study investigates the effectiveness of BERT and bi-directional LSTM models for relation prediction tasks, focusing on identifying relationships between people, locations, dates, and educational degrees mentioned in text. Through our analysis, we aim to provide insights into the capabilities and limitations of BERT-based and LSTM-based models for relation Classification.

2. Research Question and Methodology

2.1. Research Question

How can bi-directional LSTM and BERT models improve the accuracy of relation classification tasks?

Does Knowledge Graph (KG) techniques represent the relation between subjects and objects that extracted from texts?

2.2. Objectives

1. Investigate the performance of LSTM and bi-directional LSTM models in relation classification.
2. Evaluate the effectiveness of BERT models with different layers in the same task.
3. Compare and analyze the results to understand the potential and limitations of both approaches.

4. Using Knowledge Graph (KG) techniques to represent the relation between subjects and objects by using provided predicates.

2.3. Methodology and Model Architectures:

LSTM:

Long Short-Term Memory (LSTM) networks are a powerful type of recurrent neural network well-suited for tasks involving sequences, especially natural language. Unlike simpler models, LSTMs can remember information for longer periods, allowing them to capture complex relationships between words even when they are far apart in a sentence.

This paper proposes an LSTM-based model for relation classification. The model consists of several key components:

- **Input Layer**: similar to other models, the input layer processes sentences with a fixed maximum length.
- **Embedding Layer:** This layer converts raw text into numerical representations that capture word meaning and context.
- **LSTM Layers:** The core of the model, 2 LSTM layers with 128 units capture long-range dependencies within the sequence.
- **Dropout Layer:** This layer randomly drops a certain percentage of neurons during training to prevent overfitting.
- **Output Layer**: The final layer uses a linear transformation (`nn.Linear`) to map the hidden state from the LSTM layer (which captures sequence information) to the desired output dimension (5 in our case).

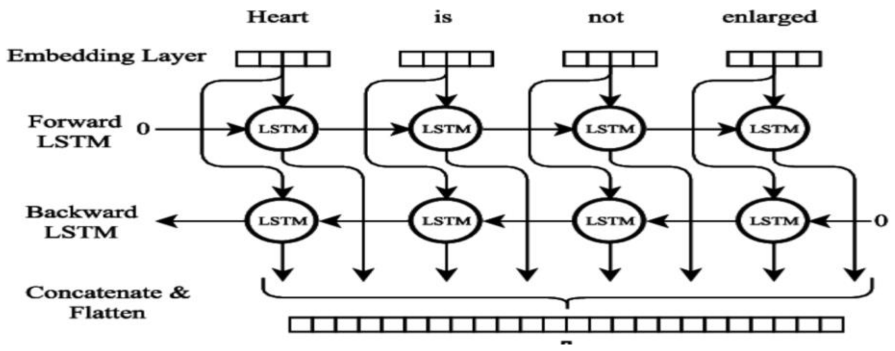
This architecture, particularly the LSTM's memory capabilities, allows the model to identify subtle relationships in text, making it a valuable tool for relation classification and relation extraction tasks that involve identifying relationships between entities in text.

Bidirectional LSTM (BiLSTM)

Bidirectional LSTM or BiLSTM is a term used for a sequence model which contains two LSTM layers, one for processing input in the forward direction and the other for processing in the backward direction. The intuition behind this approach is that by processing data in both directions, the model is able to better understand the relationship between sequences (e.g. knowing the following and preceding words in a sentence).

The model consists of several key components:

- **Embedding Layer:** Similar to the previous model, this layer remains the same, converting integer word indices to dense embedding vectors.
- **Bi-Directional LSTM Layer:** This is the key difference compared to the standard LSTM model. Here, `bidirectional=True` is set in the `nn.LSTM` constructor. This creates a Bi-directional LSTM layer that processes the sequence in both forward and backward directions simultaneously. Their hidden states are concatenated along the feature dimension (dim=1) to create a single vector
- **Dropout Layer :** Same as before, this layer helps prevent overfitting.
- **Output Layer (`self.fc`):** The final layer is modified here. Since a Bi-directional LSTM outputs two hidden states from both the forward and backward directions, the input dimension to this layer becomes `hidden_dim * 2`. This allows the model to capture information from both directions of the sequence before making a prediction.



BERT Architecture

BERT (Bidirectional Encoder Representations from Transformers), a leading language representation model created by Google. BERT employs a transformer architecture, allowing it to capture contextual information from both the left and right sides of a word in a sentence. This bidirectional context encoding is particularly advantageous for tasks such as relation classification, where understanding the surrounding context is vital for precise classification.

For our implementation, we utilized the “bert-base-uncased" version of BERT, which includes 12 transformer layers and 768 hidden units. We enhanced the pre-trained BERT model by adding a classification layer to perform relation classification. This classification layer maps the encoded input representation to the correct relation label using a softmax activation function. The number of labels was set to five to match the five relation types in our dataset.

The BERT model consists of several key components:

- **BERT Layer:** We use the pre-trained "bert-base-uncased" model to obtain contextual embeddings of the input text.
- **ReLU Activation:** A ReLU activation function introduces non-linearity.
- **Classification Layer:** A final linear layer maps the transformed features to the desired number of labels (5 in this case).

BERT_Larg Architecture

Our enhanced BERT-based classifier architecture incorporates additional layers for improved feature extraction and classification. The structure includes:

- **BERT Layer:** We use the pre-trained "bert-base-uncased" model to obtain contextual embeddings of the input text.
- **Dropout Layer:** A dropout layer with a dropout rate of 20% is applied to the BERT output to prevent overfitting.
- **Fully Connected Layer:** A linear layer reduces the dimensionality from 768 to 64.
- **ReLU Activation:** A ReLU activation function introduces non-linearity.
- **Classification Layer:** A final linear layer maps the transformed features to the desired number of labels (5 in this case).

3. Experimental Results

3.1. Dataset:

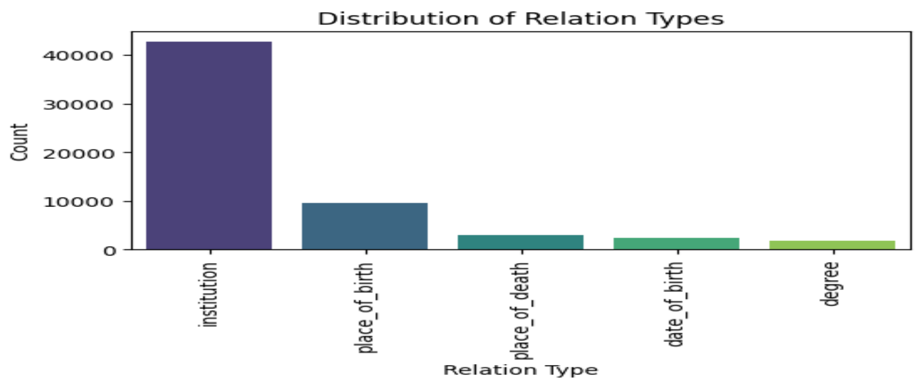
The datasets used in this project included information about places of death, dates of birth, education degrees, institutions, and places of birth.

3.2. Data Cleaning and preprocessing:

- The provided datasets were loaded and cleaned to ensure consistency and accuracy.
- JSON files such as `place_of_death.json`, `date_of_birth.json`, `education-degree.json`, `institution.json`, and `place_of_birth.json` were utilized. These files are combined for further analysis.
- For Implementing the Knowledge Graph Technics, relevant information was extracted from these datasets and formatted into triples (subject, predicate, object).

Addressing Class Imbalance

As illustrated in the following plot, the dataset exhibits significant class imbalance, with the 'institution' class representing approximately 70 percent of the data. This imbalance can lead to challenges in model training. To address these issues, we implemented undersampling techniques.



This approach resulted in a balanced dataset with 9250 texts, evenly distributed across five classes, each containing 1850 texts.

To facilitate model training, we encoded the relation labels into numerical format. This step involved mapping relation types to numerical labels using a label encoder. After the initial data manipulation phase, we proceeded with the data preprocessing stage, which involved tokenizing and preparing the text data for ingestion into the BERT model.

3.2. Training and Evaluation:

All models were trained on the preprocessed data using appropriate training parameters. Performance was evaluated using standard metrics such as precision, recall, and F1-score for both models.The model was trained using the AdamW optimizer with different learning rate. The training loop iterated over multiple epochs, with the model's performance monitored on the validation set.

3.2.1. Evaluation Metrics

In this section, the performance of the bi-directional LSTM and BERT models was evaluated using the following metrics:

F1 Score (Fc)

The F1 score, denoted as Fc, is a metric that combines precision and recall to provide a balanced evaluation of the model's performance. It is calculated using the formula:

$$Fc = 2 \cdot Pc \cdot Rc / (Pc + Rc)$$

where Pc represents precision and Rc stands for recall.

Macro-Averaged F1 Score (F)

The macro-averaged F1 score (F) is obtained by summing up the F1 scores for each relation type (Fc) and then dividing by the total number of relation types (|Cp|):

$$F = \Sigma c \in Cp \ Fc / |Cp|$$

where Cp represents the set of relation types and |Cp| denotes the total number of relation types. In the case of multi-class classification, we adopt averaging methods for F1 score calculation.

3.2.1. Description of Results

3.2.1.1. Hyperparameter Tuning Report for LSTM and Bidirectional LSTM Model

This report focuses on the hyperparameter tuning process for the LSTM and Bidirectional LSTM model. The aim was to evaluate the impact of different learning rates and hidden dimensions on the performance of the LSTM classifier for relation classification tasks. The model was trained for 20 epochs on the training dataset. A classification report and confusion matrix were generated for each combination of hyperparameters.

Average F1-Score for different learning rate

Model	Hidden Dimension	**Avg** F1-Score					
		Lr = 1e-05	Lr = 1e-04	Lr = 0.001	Lr = 0.005	Lr = 0.01	Lr = 0.05
LSTM	**128**	0.56	0.68	0.69	0.72	0.72	0.07
	256	0.52	0.69	0.75	0.74	0.71	0.07
BiLSTM	**128**	0.69	0.80	0.79	0.75	0.61	0.07
	256	0.70	0.82	0.81	0.72	0.07	0.07

The results indicate that the performance of both the LSTM and BiLSTM models varies significantly with changes in learning rate and hidden dimension. Key observations include:

- **Learning Rate:**
  - For both LSTM and BiLSTM models, a learning rate of 0.05 consistently resulted in poor performance, as indicated by the low F1-Scores (0.07).
  - The optimal learning rate for the LSTM model with a hidden dimension of 128 was found to be 0.01, achieving an F1-Score of 0.73. While the hidden dimension of 256 performed optimally at a learning rate of 0.001, achieving an F1-Score of 0.75.
  - The optimal learning rate for the BiLSTM model with a hidden dimension of 128 was 0.001, achieving the highest F1-Score of 0.79. While a hidden dimension of 256 yielded better results for the BiLSTM model, with an F1-Score of 0.81 at a learning rate of 0.001.

Reasons for Poor Performance with Learning Rate of 0.05:

- **Learning Rate Sensitivity:** Even small increases can push the model into instability, causing oscillation or divergence during training.
- **Non-Linear Interactions:** Higher learning rates might interact poorly with other hyperparameters, disrupting the learning process.
- **Overfitting to Noise:** A higher learning rate can cause the model to overfit to noise rather than learning true patterns.
- **Exploding Gradients:** LSTMs and BiLSTMs can still suffer from exploding gradients at higher learning rates.

The hyperparameter tuning process provided valuable insights into the performance dynamics of the LSTM and BiLSTM models for relation classification tasks. The findings underscore the importance of selecting appropriate hyperparameters to enhance model performance. Specifically, the BiLSTM model with a hidden dimension of 256 and a learning rate of 0.001 demonstrated the best performance. In General, **t**he BiLSTM model generally outperformed the LSTM model across different learning rates and hidden dimensions.

3.2.1.2. Hyperparameter Tuning Report for BERT and BERT_Large Model

This report focuses on the hyperparameter tuning process for the BERT and BERT_Large model. The aim was to evaluate the impact of different learning rates and two Epoch numbers (4 and 5) on the performance of the models for relation classification tasks. A classification report and confusion matrix were generated for each combination of hyperparameters.

Average F1-Score for different learning rate and two Epochs

Model	Epoch number	**Avg** F1-Score					
		Lr = 1e-05	Lr = 1e-04	Lr = 0.001	Lr = 0.005	Lr = 0.01	Lr = 0.05
BERT	**4**	0.87	0.86	0.07	0.07	0.07	0.06
	5	0.88	0.85	0.06	0.06	0.07	0.06
BERT_Large	**4**	0.88	0.85	0.07	0.07	0.07	0.06
	5	0.89	0.77	0.06	0.07	0.07	0.06

The results indicate the performance of both the BERT and BERT_Large models varies significantly with changes in learning rate and epoch number. Key observations include:

- **Learning Rate:**
  - For both BERT and BERT_Large models, a learning rate of 0.001 consistently resulted in poor performance, as indicated by the low F1-Scores (0.06 and 0.07). The model's loss did not significantly improve, indicating a potential issue with this learning rate.
  - The optimal learning rate for both models was found to be 0.00001, achieving the highest F1-Scores across different epoch numbers (0.88 and 0.89).
- **Epoch Number:**
  - Increasing the number of epochs from 4 to 5 generally improved the performance slightly for the learning rate of 0.00001.
  - However, for higher learning rates (0.0001 and 0.001), increasing the number of epochs did not consistently improve the performance and, in some cases, resulted in lower F1-Score

The hyperparameter tuning process provided valuable insights into the performance dynamics of the BERT and BERT_Large models for relation classification tasks. The findings underscore the importance of selecting appropriate learning rates and epoch numbers to enhance model performance. Specifically, the BERT_Large model with a learning rate of 0.00001 and 5 epochs demonstrated the best performance.

Comparative Analysis of LSTM/BiLSTM and BERT/BERT_Large Models

LSTM/BiLSTM models achieved optimal performance with a learning rate of 0.001, with BiLSTM reaching an F1-Score of 0.81. In contrast, BERT/BERT_Large models performed best with a much lower learning rate of 0.00001, with BERT_Large achieving an F1-Score of 0.89.

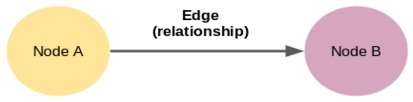
BERT/BERT_Large models generally performed better due to their ability to capture complex contextual relationships through their transformer architecture, which allows for better representation and understanding of language nuances. This highlights the need for model-specific hyperparameter tuning, with BERT models requiring lower learning rates and being a bit sensitive to epoch changes.

4. Knowledge Graph

After training and evaluating the models, the next step involved visualizing the results to provide a clear understanding of the performance and the knowledge extracted.

A knowledge graph is a way of storing data that resulted from an information extraction task. Many basic implementations of knowledge graphs make use of a concept we call triple, that is a set of three items(a subject, a predicate and an object) that we can use to store information about something. knowledge graphs effectively represent complex information.(Peng et al., 2023)

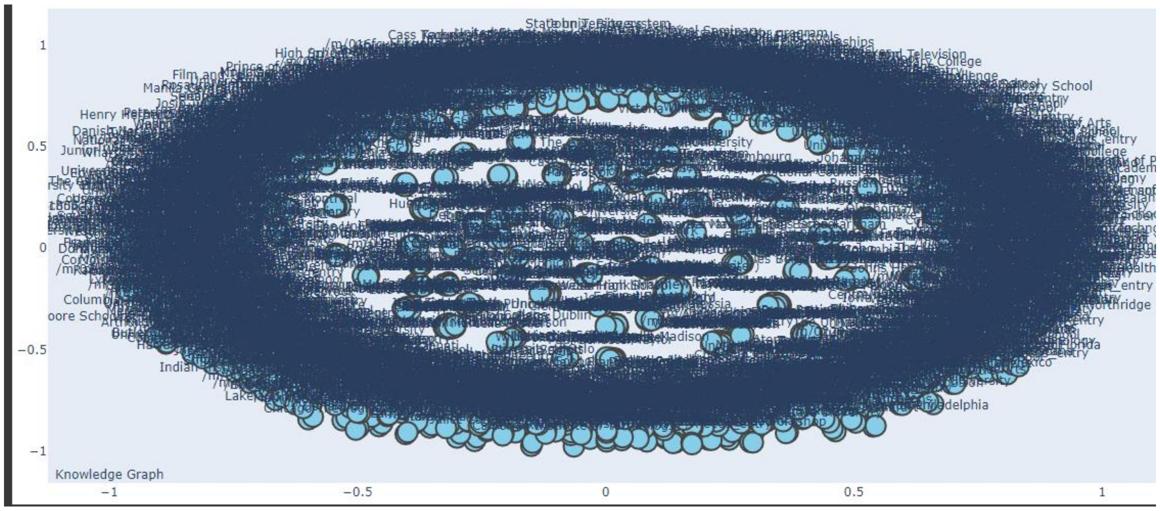
Node A and Node B here are two different entities. These nodes are connected by an edge that represents the relationship between the two nodes. Now, the following picture is the smallest knowledge graph we can build.



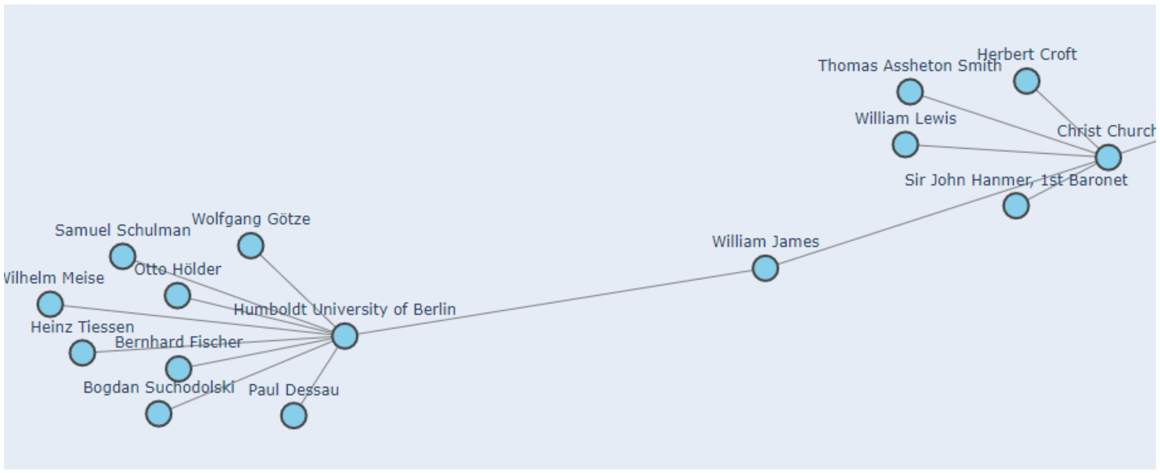
Results of Knowledge Graph:

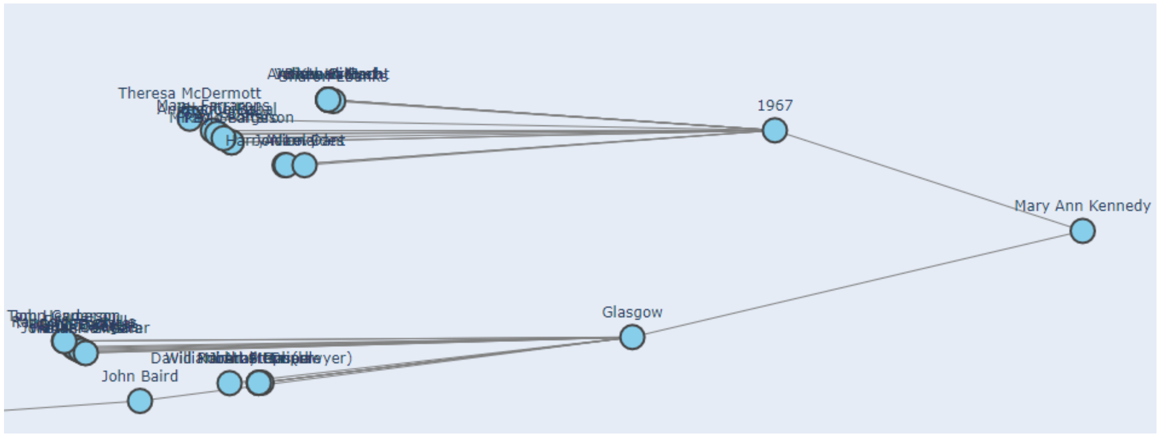
Using tools like "Networkx" library and "plotly.graph_objects" to visualize the Knowledge Graph with the classified relations. In this case, the graph nodes represented entities (e.g., persons, places, type of degrees), and the edges represented the classified relations (e.g., place of birth, date of birth, place of death, degree, institution). Graph visualized in dynamic form to be zoomed.

Metric	Value
Number of Nodes	13483
Number of Edges	9247
Number of Datasets	9250
Number of Unique Objects	4451
Number of Unique Predicates	5



By Zooming we can get more valuable information for graphs.





5. Conclusion

The hyperparameter tuning and performance evaluation of LSTM, BiLSTM, BERT, and BERT_Large models for relation classification revealed key insights. BiLSTM performed best with a learning rate of 0.001 and a hidden dimension of 256, achieving an F1-Score of 0.81. BERT_Large excelled with an F1-Score of 0.89 at a lower learning rate of 0.00001 and 5 epochs. Higher learning rates consistently resulted in poor performance across all models, highlighting the importance of proper learning rate selection.

BERT and BERT_Large models outperformed LSTM models due to their transformer architecture, which captures complex contextual relationships more effectively. These models required lower learning rates and showed greater sensitivity to epoch adjustments.

This analysis underscores the effectiveness of transformer-based models in NLP tasks and the necessity for careful hyperparameter tuning. Future work should focus on further refining these parameters to enhance model performance and robustness.

Future Work

1. Explore the integration of additional LLMs to further improve performance.
2. Experiment with different model architectures and hyperparameters to optimize results.
3. Apply the approach to other datasets and relation types to validate its generalizability.
4. Investigate the use of LSTM and BERT with different layer numbers for even better performance.

Reference

Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, Francesco Osborne: "Knowledge Graphs: Opportunities and Challenges". Mar 2023

Hendrickx, Iris, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz: "SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals". In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 33–38, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Wu, Shanchan and Yifan He: Enriching pre-trained language model with entity information for relation classification. May 2019.

https://paperswithcode.com/method/bilstm

Dataset Link:

https://code.google.com/archive/p/relation-extraction-corpus/downloads