Relation Extraction: A Comparative Study of SVM and LSTM models on SemEval-2010 Task 8

Yifan SUN, Yuxiang Sun, Murui Xiao, Gaobin Zhou

Abstract

Relation extraction (RE) is a pivotal task in natural language processing, which aims to identify semantic relationships between entity pairs in a given sentence. Our study compares two approaches for RE: a traditional Support Vector Machine (SVM) method and a deep learning approach based on Long Short-Term Memory (LSTM) networks. To further improve performance, we enhance the SVM by integrating a Hidden Markov Model (HMM) component, which better captures sequential dependencies, and we augment the LSTM architecture with a multi-head attention mechanism to allow the model to focus on the most informative parts of a sentence. We conduct our experiments on SemEval-2010 Task 8 dataset and evaluate the performance using precision, recall, and F1-score metrics. The results show that our enhancements improve upon the original methods and the LSTM-based models significantly outperform the SVM-based models. This paper presents the theoretical foundations, implementation details, and experimental results of different methods, demonstrating the effectiveness of deep learning techniques over traditional machine learning methods for the relation extraction task.

1 Introduction and Related Work

Relation Extraction was introduced by the MUC (Message Understanding Conference) in 1998 (Chinchor, 1998). It refers to the task that determining the relationship between two entities within a given sentence. For example, in the sentence "Microsoft was founded by Bill Gates," the relationship between 'Microsoft' and 'Bill Gates' is classified as "Product-Producer." RE plays a crucial role in various natural language processing (NLP) applications, such as information retrieval, knowledge base construction, and question answering (Bassignana and Plank, 2022).

Traditionally, machine learning approaches, such as Support Vector Machines (SVM) (Hong, 2005) and Kernel Methods, have been widely used for RE. These methods rely on manually engineered features to classify relationships between entities. However, while SVM-based approaches perform well on smaller datasets, they struggle with complex sentence structures and fail to capture intricate sequential dependencies. To address these limitations, researchers have explored combining generative models like Hidden Markov Models (HMM) with discriminative SVMs, improving their ability to model sequential patterns while retaining classification accuracy (Nasfi and Bouguila, 2022).

In recent years, relation extraction has experienced significant advancements, transitioning from traditional statistical methods to deep learning-based approaches. For example, Recurrent Neural Networks (RNN) have been employed to capture sequential dependencies in text (Zhang et al., 2015), while Convolutional Neural Networks (CNN) automatically learn hierarchical features from raw input (Zeng et al., 2014). More recently, pre-trained Transformer-based models, such as BERT and RoBERTa, have set new benchmarks by leveraging large-scale, contextualized word representations, significantly enhancing performance on RE tasks (Devlin et al., 2019). Among deep learning techniques, RNN based Long Short-Term Memory (LSTM) networks have demonstrated notable promise in capturing contextual dependencies in text. In particular, bidirectional LSTMs—by processing input sequences in both forward and backward directions—effectively model relationships by leveraging context from both past and future tokens, thereby enhancing accuracy in relation classification tasks (Zhang et al., 2015). Nevertheless, even with these improvements, standard LSTM models continue to face challenges in capturing subtle contextual cues, especially in longer or more complex sentences.

To address these challenges, we evaluate both tra-

ditional SVM-based and LSTM-based deep learning approaches. We propose two key enhancements: first, we integrate an HMM component with the SVM approach to better capture sequential dependencies; second, we augment the LSTM architecture with a multi-head attention(HAM) mechanism (Valentino et al., 2024), enabling the model to focus on the most informative parts of a sentence.

The SemEval-2010 Task 8 dataset (Hendrickx et al., 2010) is used as the benchmark for evaluating RE systems in this study. It consists of 10,717 labeled sentences from diverse sources, such as news articles and Wikipedia, providing a comprehensive representation of real-world language. Each sentence is annotated with one of nine relation types or a 'no relation' label if no meaningful relationship is present. Importantly, the entities in each sentence are marked, enabling models to focus directly on learning the relationships between them. The dataset is balanced across relation types, ensuring that models trained on it can generalize effectively to a wide range of relationship patterns.

The structure of this paper is as follows: Section 2 outlines the methodology, including data preprocessing, the implementation of SVM and LSTM models, and the enhancements we propose. Section 3 presents experimental results and evaluation metrics, comparing the performance of different models. Finally, Section 4 concludes the paper by summarizing our key findings and discussing potential future research directions.

2 Methodology

2.1 Data Preprocessing

Data preprocessing is a crucial first step in relation extraction, aiming to convert raw textual data into a numerical format that models can process effectively. This phase includes three key procedures: data standardization, text parsing and sequence construction, and semantic representation initialization.

First, data standardization involves converting raw text into a structured format, typically JSON, where each sentence is paired with its corresponding relation label. These relation labels are then mapped to numerical identifiers using a predefined mapping table.

Next, text parsing and sequence construction transform sentences into fixed-length numerical sequences. This begins with tokenization(Kim, 2021), where sentences are split into tokens, which are mapped to numerical identifiers based on a vocabulary. Tokens not found in the vocabulary are assigned an "UNK" identifier. Sequences are then padded or truncated to a predefined maximum length, with a mask sequence indicating valid tokens versus padding.

Finally, semantic representation initialization(Chen, 2022) involves constructing word embeddings for each token using a pre-trained word embedding model. These embeddings are stored in a matrix, with special vectors assigned to tokens like Padding, Unknown, and Entity Boundary markers.

2.2 Support Vector Machine(SVM)

Support Vector Machine (SVM) is a supervised learning algorithm used for classification, aiming to find the optimal hyperplane that maximizes the margin between different classes. Given a training dataset:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

Among them: $x_i \in \mathbb{R}^d$ is the eigenvector of the ith sample, $y_i \in \{-1, +1\}$ is the class label of the sample, n is the number of samples, d is the characteristic dimension of each sample.

The goal of SVM is to find a hyperplane that maximizes the margin (i.e., the distance between the hyperplane and the nearest sample point). This hyperplane can be expressed as:

$$w^T x + b = 0$$

where $w \in \mathbb{R}^d$ is a weight vector perpendicular to the hyperplane.

Margin refers to the distance from the hyperplane to the nearest data point. The point closest to the hyperplane is called a support vector. For a given data point (x_i, y_i), we want to ensure that:

$$y_i(w^T x_i + b) \geq 1$$

which leads to the optimization problem:

$$\text{minimize } \frac{1}{2}\|w\|^2$$

The constraint conditions are:

$$y_i(w^T x_i + b) \geq 1, \quad \forall i = 1, 2, \cdots, n$$

For non-linearly separable data, SVM uses the kernel trick to project data into a higher-dimensional space via a kernel function K(xi,xj),

enabling linear separation in the transformed space. The final decision function is:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, \mathbf{x}) + b\right),$$

where the kernel function is defined as:

$$K(x_i, x_j) = x_i^T x_j.$$

This enables SVM to efficiently classify nonlinearly separable data by learning a linear hyperplane in a high-dimensional space.

2.3 Hidden Markov Model (HMM)

A Hidden Markov Model (HMM) is a probabilistic model used for sequential data, where hidden states generate observable outputs. It is widely applied in NLP, including part-of-speech (POS) tagging. HMM consists of two basic steps:

State Transition Probabilities: The probability of a transition from one hidden state to another. We represent the transition matrix by $A = \{a_{ij}\}$, where a_{ij} represents the probability of moving from state i to state j. The transition probabilities satisfy the condition:

$$\sum_{j=1}^{N} a_{ij} = 1, \quad \forall i \in \{1, 2, \ldots, N\}$$

Observation Probabilities: The probability of generating an observation in a hidden state. We use $B = \{b_i(o)\}$ to represent the probability of observation, where $b_i(o)$ represents the probability of observing o in state i. The observation probabilities satisfy the condition:

$$\sum_{o \in O} b_i(o) = 1, \quad \forall i \in \{1, 2, \ldots, N\}$$

Finally, the trained HMM model is used to extract the part-of-speech labels of each sentence. These labels can eventually input as features into the SVM model for subsequent relation extraction tasks.

2.4 LSTM with multi-head attention

We employ a deep learning model based on LSTM networks and a multi-head attention mechanism (Cao and Shao, 2024) to extract relationships between entities from the SemEval 2010 Task 8 dataset. This model comprises three main components: the embedding layer, the LSTM layer, and the multi-head attention mechanism. This architecture effectively captures both local and global text dependencies, enabling the model to better understand the relationships between entities. Figure 1 illustrates the structure of the entire model.

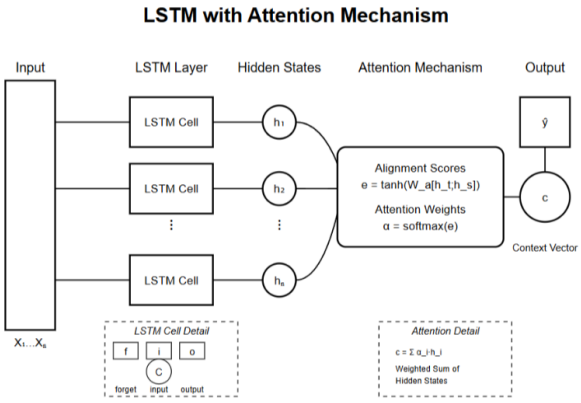

Figure 1: The structure of LSTM with Multi-head attention mechanisms

The initial component is the embedding layer, which converts input text into high-dimensional vectors. At this stage, we use a pre-trained GloVe embedding(Parmar, 2024) with 100 dimensions to initialize the embedding matrix, providing a rich semantic foundation. We then assign unique identifiers to special tokens such as PAD, UNK, <e1>, and <e2>. These entity position markers help the model recognize target pairs in sentences (e.g., "The <e1>virus</e1> replicates in <e2>cells</e2>"). To enhance generalization during fine-tuning, we apply a dropout rate of 0.3 at this stage(Lin, 2023).

The LSTM layer receives the embedding layer's output, capturing sequential dependencies and contextual information. This layer is configured with 100 hidden units to balance efficiency and performance. To handle varying sequence lengths, we use `pack_padded_sequence` and `pad_padded_sequence` operations along with appropriate masking, ensuring the model focuses only on valid tokens during training. A dropout rate of 0.3 is applied to the LSTM output to prevent overfitting.

On top of the LSTM layer, we add a multi-head attention mechanism to capture different aspects of the sequence. Here, attention scores are computed using the scaled dot-product method, with explicit masking to minimize the influence of padding tokens. Outputs from multiple attention heads are combined via linear transformations, producing a

final sequence that retains both local and global context.

Finally, the attention-processed sequence is condensed into a fixed-length vector through average pooling. This vector is projected into the classification space using a linear transformation layer, and the Softmax activation function(Lu and et al., 2024) generates a probability distribution over potential output classes. Additionally, a dropout layer with a rate of 0.5 is applied before the linear layer to reduce overfitting risk. The model weights are initialized using the Xavier method(Glorot and Bengio, 2010), and biases are set to zero to accelerate training convergence and enhance overall stability.

3 Result and Discussion

To evaluate the performance of the models, we used the F1-score as the main criterion, which balances both precision and recall. This metric is particularly suitable for relation extraction tasks, where class imbalance can make accuracy unreliable. A higher F1-score indicates better generalization and effective identification of relevant relations.

	SVM		**SVM-HMM**	
	Train	**Test**	**Train**	**Test**
Precision	0.7777	0.6296	0.7788	0.6397
Recall	0.7708	0.6467	0.7782	0.6485
F1-score	0.7626	0.6333	0.7668	0.6402

Table 1: Performance comparison between SVM and SVM-HMM

We trained the SVM model on the SemEval-2010 Task 8 dataset using TF-IDF vectorization (n-gram range: 1–3, max features: 4000) to represent sentence text features. Class imbalance was addressed with automatic class weighting. The SVM model achieved an F1-score of 0.7626 (train) and 0.6333 (test), indicating a moderate decline in generalization. To enhance performance, we introduced HMM to extract part-of-speech (POS) sequences, encoded them using CountVectorizer, and combined them with TF-IDF features. This improved the test F1-score to 0.6402, suggesting that POS features helped capture additional linguistic patterns, though the gain was limited, highlighting SVM's difficulty in modeling sequential dependencies in relation extraction.

To train the LSTM model, we utilized the Stochastic Gradient Descent (SGD)(Lopez and Victor, 2020) optimizer with a learning rate of 1 and employed L2 regularization to manage model complexity and prevent overfitting. The training involved 10 epochs, each with a batch size of 10. Training data was randomly shuffled before each epoch to enhance generalization.The F1-score of the LSTM model reaches 0.9 and 0.8 on the training set and test set, respectively, and improves to 0.93 and 0.82 after Multi-head Attention treatment, which suggests that the introduction of Multi-head Attention can improve the performance of the LSTM model.

	LSTM		**LSTM-MHA**	
	Train	**Test**	**Train**	**Test**
Precision	0.9076	0.7883	0.9305	0.8143
Recall	0.8912	0.8064	0.9237	0.8185
F1-score	0.8993	0.7940	0.9269	0.8152

Table 2: Performance comparison between LSTM and Multi-head Attention LSTM

In comparing the SVM and LSTM models for relation extraction task, we found that SVM struggled with generalization, showing a notable drop in performance from the training to the test set. While adding HMM for POS sequences slightly improved performance, SVM still had difficulty handling sequential dependencies. In contrast, the LSTM model performed better overall, with a strong ability to capture long-range dependencies. The addition of Multi-head Attention further enhanced its performance, demonstrating improved generalization and effectiveness in relation extraction tasks.

4 Conclusion

The results of this study suggest that LSTM models, particularly when enhanced with Multi-Head Attention, significantly outperform traditional SVM-based models, both in terms of accuracy and generalization. While the SVM model with HMM preprocessing offered some improvement, it still struggled with overfitting, highlighting the limitations of this approach for tasks requiring the capture of sequential dependencies. The LSTM model, on the other hand, demonstrated better generalization capabilities, and the incorporation of Multi-Head Attention further boosted its performance, making it more effective for complex relation extraction tasks. Future work could explore further enhancements to LSTM models, including experimenting with Transformer-based architectures and other attention mechanisms, to continue improving performance on challenging relation extraction tasks.

References

E. Bassignana and B. Plank. 2022. What do you mean by relation extraction? a survey on datasets and study on scientific relation classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 67–83.

X. Cao and Q. Shao. 2024. Joint entity relation extraction based on lstm via attention mechanism. *Arabian Journal for Science and Engineering*, 49(3):4353–4363.

H. et al. Chen. 2022. Semantic-aware dense representation learning for remote sensing image change detection. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–1.

N. Chinchor. 1998. Overview of muc-7/met-2. https://www-nlpir.nist.gov/related_projects/muc/.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *In Proceedings of NAACL-HLT 2019*, pages 4171–4186.

X. Glorot and Y. Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, 9:249–256.

I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. O Séaghdha, S. Padó, and K. Torisawa. 2010. 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.

G. Hong. 2005. Relation extraction using support vector machine. In *Proceedings of the Second International Joint Conference on Natural Language Processing: Full Papers*.

G. et al. Kim. 2021. Enhancing korean named entity recognition with linguistic tokenization strategies. volume 9, pages 1–1.

H. et al. Lin. 2023. Learning rate dropout. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):9029–9039.

D. Lopez and N. Victor. 2020. sl-lstm: A bi-directional lstm with stochastic gradient descent optimization for sequence labeling tasks in big data. *International Journal of Grid and High Performance Computing*, 12(3):1–16.

J. Lu and et al. 2024. Softmax-free linear transformers. *International Journal of Computer Vision*, 132(8):3355–3374.

R. Nasfi and N. Bouguila. 2022. Sentiment analysis from user reviews using a hybrid generative-discriminative hmm-svm approach. In *In Proceedings of the Joint IAPR International Workshops on Structural, Syntactic, and Statistical Pattern Recognition, and 22nd International Conference on Pattern Recognition (SSPR & ICPR)*, pages 74–83. Springer.

D. et al. Parmar. 2024. Cognitive engagement detection of online learners using glove embedding and hybrid lstm. In *In Generative Intelligence and Intelligent Tutoring Systems, Part II, ITS 2024*, pages 15–26. Springer.

M. Valentino, D.S. Carvalho, and A. Freitas. 2024. Multi-relational hyperbolic word embeddings from natural language definitions.

D. Zeng, K. Liu, S. Lai, G. Zhou, and J. Zhao. 2014. Relation classification via convolutional deep neural network. In *In Proceedings of COLING 2014*, pages 2335–2344.

S. Zhang, D. Zheng, X. Hu, and M. Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 73–78.