AN IMPROVED NLP FOR SYNTACTIC AND SEMANTIC MATCHING USING BIDIRECTIONAL LSTM AND ATTENTION MECHANISM

Fadya Abbas

Department of Computer Engineering, Nahrain University, Iraq

ABSTRACT

Dealing with extensive amounts of textual data requires an efficient deep learning model to be adapted. However, the following reasons; the highly ambiguous and complex nature of many prosodic phrasing also enough dataset suitable for system training is always limited, cause big challenges for training the NLP models. This proposed conceptual framework aims to provide an understanding and familiarity with the elements of modern deep learning networks for NLP use. In this design, the encoder uses Bidirectional Long Short-Term Memory deep network layers, to encode the test sequences into more context-sensitive representations. Moreover, the attention mechanism is mainly used to generate a context vector that is determined from distinct alignment scores at different word positions, hence, it can focus more on a small words' subset. Hence, the attention mechanism improved the model data efficiency, and the model performance is validated using an example of data sets that show promise for a real-life application.

KEYWORDS

NLP, Deep Learning, LSTM, Attention Mechanism, Data Efficiency.

1. INTRODUCTION

Syntactic and semantic analysis is becoming more crucial with time as it can provide valuable information needed in many wide-world applications. Several Natural Language Processing (NLP) based syntactic and semantic analysis methods that trained using the following deep learning models: RNN, GRU and LSTM models [1]. Bidirectional LSTM (BiLSTM) deep neural network layers have been used more in NLP algorithms, particularly in the Encoder stage. Basically, the BiLSTM layers are implemented in a Bidirectional manner, to form two learning models in different directions [2]. The main advantage of the BiLSTM layer is to encode the text sequences into more context-sensitive representations. Recently, attention mechanism designs led to the development of modern NLP architectures. [3] This research conducted three syntactic experiments. Also, their data is associated with different percentages of noise. These experiments use the following three models; the conventional LSTM, bi-directional LSTM, and bi-directional LSTM with Conditional Random Field (CRF) which has given the best performance among the other two. [4] This research also presents an architecture for argument classification that uses Bidirectional LSTM and CRF decoding for finding optimal sequences. This method is based on the combination of syntactic features and external word representations from FastText, where the FastText is used to obtain better results for words that do not exist in the dictionary.

The Attention mechanism essentially generates a context vector that is computed from various alignment scores at various word positions, so it can focus on a small words' subset. It weighs all inputs individually that are fed to the decoder to create the target sequence. This leads to a better contextual understanding resulting in a maximum prediction score in target sequence generation. Also, attention mechanism is used widely in the language translation fields [5].

However, these algorithms still need to be implemented in the syntactic and semantic analysis in terms of data efficiency perspective. Hence, the proposed NLP algorithm uses the attention layer to extract only the valuable data and remove the others; therefore, it can improve the training limitation problem and leverage the model into more data-efficient.

Figure. 1 demonstrates the overall system architecture, particularly, it shows that the input clauses are encoded into sequences of distributed vectors in the Encoder stage. The learnable parameters; E, W, M, and W' are the adjustable weights between the model hidden layers. The final layer of the Encoder sends the output Y_{t+1} information to the Decoder after an extensive processing of data dependencies in the attention layer (with the output h_{t+1}). Finally, the Decoder can analyse the entire contextual understanding of all the previous words via probability distribution of the Softmax function. Combining each hidden layer's output, the weight matrices, and bias (b) can mathematically be expressed as follows:

$$\text{Encoder: } Y_{t+1} = Softmax(E_{t+1} . x_{t+1} + W_t . Y_t + b_{t+1}) \quad (1)$$

$$\text{Decoder: } O_{t+1} = Softmax(M_{t+1} . h_{t+1} + W'_{t-1} . O_t + b_{t+1}) \quad (2)$$

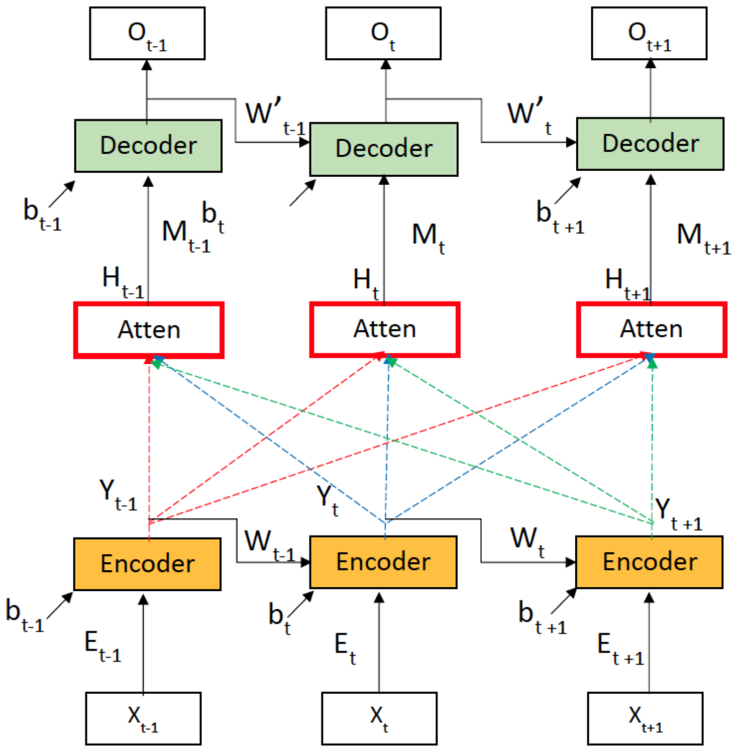

Figure 1. Overall NLP architecture.

Where the Softmax is the activation function at the neural network layer output. With attention assistance, the Decoder can determine the extent of the match between the t-th input and the corresponding t-th output information. More details on the main parts and the pipeline data processing of the above diagram are clarified below.

2. Architecture Details

The following Architecture-based approaches are organized into four main levels using text sequences' analysis. Each level constructs a model and shows the algorithm that uses syntactic elements of a text and how to initiate the relevant learning weights to eventually ends with a prediction category for the final text sequence patterns.

2.1. Word Embeddings and Tokenization

The text is processed in this model and represented as a sequence of embedding tokens to express the semantic features of the sequences; given the text X = (x_1; x_2; : : : ; x_n) in which consisting of n clauses, and with each clause, xt =(w_1;w_2; : : : ;w_m) containing m words. First, we obtain a hidden representation and the tokenization process of the sequences using the pre-trained model called Bidirectional Encoder Representations from Transformers (BERT); T_t = BERT(x_t). In this way, the input clauses are encoded into a sequence of the distributed vectors T = [T_1; T_2; : : : ; T_n]. In BERT, four types of embedding: token embedding, segment embedding, position embedding, and topic embedding. The token embedding performs the embedding vector of each word, segment embedding is utilized to distinguish sentences, position embedding learns embedding at each word position in order to represent the sequences' order information, finally, the topic feature embedding is used to capture the underlying topic information [2, 6]. With these mentioned combined features, it can generate different embeddings for polysemous words and can model words and context by characters which leads to better contextual management, especially with the words that have multiple grammatical structures.

2.2. The Encoder - BiLSTM Layers Implementation

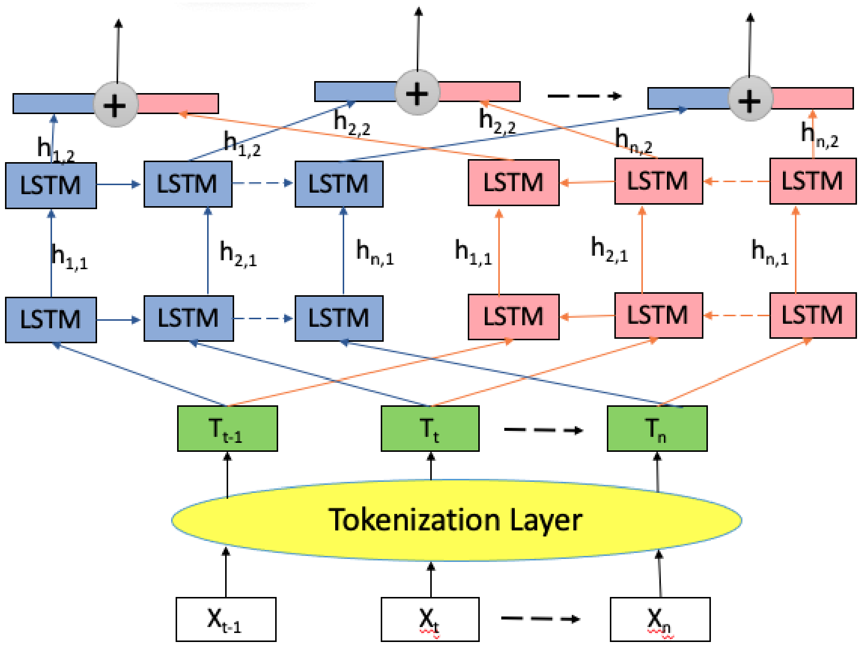

Figure 2. Encoder architecture - using BiLSTM.

Recently, the LSTM architecture has already offered great prediction results in many engineering applications. In this work, BiLSTM is used in the Encoder stage to enable additional training by crossing over the input data sequences twice. This can be happened by including two LSTM layers, one takes the forward direction sequences, and the other takes the backwards direction sequences, which offers better predictions. In this way, The approach can capture contextualized word representations in a clause. So, BiLSTM model leverages the Encoder task in extraction of the contextual word representations. The goal of BiLSTM is to model (1) complex semantical and syntactical characteristics of words (2) lexical ambiguity or polysemy, words with similar pronunciations could have different meanings at different locations or contexts [7].

At time step t-1 the forward language model can predict the next token T_t for the given previous tokens of the input sequence using the joint probability distribution as in (3). For the backward direction it is following the same manner except it is in a reversed order as shown in (4) [8, 9]:

$$p(T_1, T_2, .., T_n) = \prod_{i=1}^{N} p(T_i \mid T_1, T_2, .., T_{i-1}) \text{Forward direction} \quad (3)$$

$$p(T_1, T, .., T_n) = \prod_{i=1}^{N} p(T_i \mid T_{i+1}, T_{i+2}, .., T) \text{Backward direction} \quad (4)$$

Hence, the two LSTMs can process sentence inputs twice in reversed directions to encode the input sequence T into more context-sensitive representations. For each token representation $T_i$the bi-directional vector representations can be computed by their two hidden layers$\vec{h}_{ij}$and $\overleftarrow{h}_{ij}$, in which their weights can be updatedusing μ of each LSTM model as

$$\overrightarrow{h_{ij}} = \overrightarrow{LSTM}\left(T_i, \overrightarrow{h_{i-1,j}};\ \mu_{LSTM}\right) \text{Forward direction} \quad (5)$$

$$\overleftarrow{h_{ij}} = \overleftarrow{LSTM}\ (T_i, \overleftarrow{h_{i+1,j}};\ \mu_{LSTM}) \text{Backward direction} \quad (6)$$

Where $\vec{h}_{i-1,j}$and $\overleftarrow{h}_{i+1,j}$ are the forward and backward LSTM hidden layers respectively. By concatenating these two directional outputs the final representation to be sent to the attention layer is as follows:

$$Y_{t+1} = \left[\overrightarrow{h_{ij}} + \overleftarrow{h_{ij}}\right] \quad (7)$$

2.3. The Attention Layer

The main objective of the attention layer is to detect long-range dependency between the word pairs in a sentence using the attention weights. Therefore, the model can capture pair-wise relations of the input tokens in a sequence. So, It sets large weights on the important tokens to the word. In the other words, the attention layer enhance the contextualization by generating attention vectors, that can determine the relevance of the t-th word in a sequence that concerning other words. To obtain the attention output, Feed Forward Neural (FFN) Network receives the attention vectors and outputs the attention’s result to the Decoder's model. In fact, the Encoder output and Decoder input embeddings are both fed to the attention to performs attention between the them. This obtains the relevance of the input's tokens concerning its targets tokens as the Decoder determines the actual vector representation between the mapping target and the source [3].

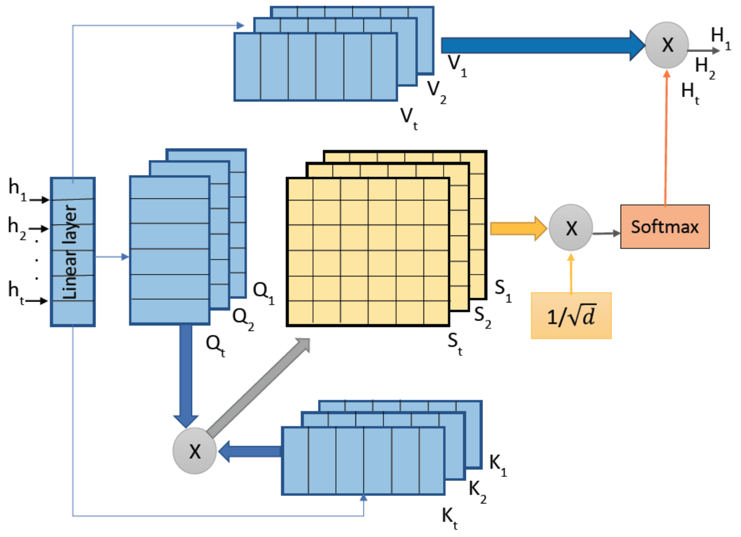

Figure 3. The attention mechanism.

It is worth noting that the attention layer is formed by multihead attentions, which contains of t attention heads, each learns different representation in a sequence. The main task of multihead attention is to map the hidden state sequence received from the BiLSTM into t different query, key and value matrices via a linear projection. Therefore, for the t-th head, the queries, keys and values can be represented by Q, K and V matrices respectively. Finally, the context vectors for each attention head (H_t) can be calculated using the following formulation [10]:

$$H_t = Attention(\, Q\, W_t^Q, K\, W_t^K, V\, W_t^V) \qquad (8)$$

$$H_t = Softmax\left(\frac{(Q\, W_t^Q)\,(K\, W_t^K)}{\sqrt{r}}\right) V . W_t^V \, (9)$$

Where the W^Q_t, W^K_t, and W^V_t are the weights of Q, K, and V matrices respectively. The softmax function is used to find the probability of the matrices correlation, where $\sqrt{r}$ is just to scale the final attention score.

Multihead attention enhances the model's capability to attend sequence parts from a different perspective by implementing attention parallelly multiple times. This performs a self-attention vector for each token by weigh the word with higher than others by the high resultant dot product.

This attention mechanism will be implemented in parallel for multiple times is computed which result in multiple attention vectors for each token to determine the attention vector. Finally, the resulting heads or attention outputs are concatenated and then transformed to the next LSTM layer (in the Decoder).

2.4. The Decoder

After capturing the information using the previous multihead attention layer, the sequence representations are sent to the next stage which is the Decoder. In fact, multihead attention can help the Decoder in better learning the soft alignment between the summary and the source document. The Decoders predicts the final summary representation also learns the final objective which is maximizing the likelihood of the conditional probability. Therefore, as shown in the figure. 1 the Decoder is also using LSTM Layer in order to predict the final summary representation. This layer is implemented to model the output distribution over the class tags then generates an output sequence of predicted tags (S_t) [7, 8]:

$$S_{t+1} = LSTM(H_t, S_t;\ \theta_{LSTM}) (10)$$

Subsequently, the output of Decoder's LSTM is sent to the Softmax layer over tag vocabularies. The Softmax layer determines the normalized probability distribution over thw labels of the possible phrase for each word:

$$p^{pred}(O_t = l_t | S_{1:N} = \frac{\exp(W_t S_t + b_t)}{\sum_{t=1}^{N} W_t S_t + b_t}) \qquad (11)$$

Where p^{pred} represents the possibility that the t-th token’s label is in the label set, O_t represents the output of the t-th token. W_t and b_t are referred to the weight vectors and bias vectors respectively. l_t is the ground truth of the clause S_t for the tag prediction, and N denotes to the total number of tags.

This model is trained to maximize the log-likelihood of a tag prediction as an objective function in which the following loss function can be determined:

$$Loss_t = \sum_{k=1}^{K} p_k^{real} * \log(p_k^{pred}) (12)$$

$Loss_t$ denotes to the loss of the t-th token. K is the length of label set. p_k^{real} denotes the real possibility that the t-th token‘s label in the label set is close or equal to 1 if the label is the real label, otherwise is 0.

3. Initial Results

The proposed algorithm applied to the training data set represented by four chapters of the Moby Dick book. The test procedure passes 25 words as test data to the NLP algorithm and the algorithm predicts the 26th word. The vocabularies of all words that occurred less than five times in the training data set have been discarded.

Table 1. Accuracy of syntactic and semantic matching.

NLP Algorithm	Syntactic [%]	Semantic [%]	Total accuracy [%]
Without Attention Mechanism	59	54	57
With Attention Mechanism	61	58	60

Table. 1. shows the precision syntactic and semantic matching. It has experimented on ten sentences frequency. As it can be seen from the table, the attention mechanism can improve the syntactic and semantic word relationships accuracy prediction.

4. Conclusions

The main key contribution of this work is to obtain a syntactic and semantic prediction. The proposed NLP architecture is represented by combining two powerful deep learning models; BiLSTM and attention mechanism, showing how to train distributed representations of words and phrases to make precise analogical reasoning possible. It is successfully trained the models on several terms and phrases. With more improvements to the current model that the author planned to achieve in future work, the technique can be used to train the large-of-words models and for the syntactic and semantic prediction fields. Thanks to the attention mechanism for its computationally efficient architecture to deal with critical training data set.

REFERENCES

[1] Q. A. Shreda and A. A. Hanani, "Identifying Non-functional Requirements from Unconstrained Documents using Natural Language Processing and Machine Learning Approaches," in IEEE Access, doi: 10.1109/ACCESS.2021.3052921.

[2] Z. Ke, J. Sheng, Z. Li, W. Silamu and Q. Guo, "Knowledge-Guided Sentiment Analysis Via Learning From Natural Language Explanations," in IEEE Access, vol. 9, pp. 3570-3578, 2021, doi: 10.1109/ACCESS.2020.3048088.

[3] R. Liu, B. Sisman, F. Bao, J. Yang, G. Gao and H. Li, "Exploiting Morphological and Phonological Features to Improve Prosodic Phrasing for Mongolian Speech Synthesis," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 274-285, 2021, doi: 10.1109/TASLP.2020.3040523.

[4] A. Jettakul, C. Thamjarat, K. Liaowongphuthorn, C. Udomcharoenchaikit, P. Vateekul and P. Boonkwan, "A Comparative Study on Various Deep Learning Techniques for Thai NLP Lexical and Syntactic Tasks on Noisy Data," 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2018, pp. 1-6, doi: 10.1109/JCSSE.2018.8457368.

[5] D. Vasić and M. K. Vasić, "Syntax-aware Neural Semantic Role Labeling for Morphologically Rich Languages," 2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2020, pp. 1-6, doi: 10.23919/SoftCOM50211.2020.9238179..

[6] Z. Kong, C. Yue, Y. Shi, J. Yu, C. Xie and L. Xie, "Entity Extraction of Electrical Equipment Malfunction Text by a Hybrid Natural Language Processing Algorithm," in IEEE Access, vol. 9, pp. 40216-40226, 2021, doi: 10.1109/ACCESS.2021.3063354.

[7] C. Fan, C. Yuan, L. Gui, Y. Zhang and R. Xu, "Multi-Task Sequence Tagging for Emotion-Cause Pair Extraction Via Tag Distribution Refinement," in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 29, pp. 2339-2350, 2021, doi: 10.1109/TASLP.2021.308983.

[8] Y. Hu, X. Qiao, L. Xing and C. Peng, "Diversified Semantic Attention Model for Fine-Grained Entity Typing," in IEEE Access, vol. 9, pp. 2251-2265, 2021, doi: 10.1109/ACCESS.2020.3046787.

[9] A. A. Syed, F. L. Gaol and T. Matsuo, "A Survey of the State-of-the-Art Models in Neural Abstractive Text Summarization," in IEEE Access, vol. 9, pp. 13248-13265, 2021, doi: 10.1109/ACCESS.2021.3052783.

[10] T. Ma, Q. Pan, H. Rong, Y. Qian, Y. Tian and N. Al-Nabhan, "T-BERTSum: Topic-Aware Text Summarization Based on BERT," in IEEE Transactions on Computational Social Systems, doi: 10.1109/TCSS.2021.3088506.

AUTHOR

Fadya Abbas

Received the BSc. Degree in computer engineering from Nahrain University, Baghdad, Iraq, in 2011. Her research interest in Deep Learning includes Natural Language Processing and Computer Vision and their applicability in different real-life applications. She is currently based in Edmonton, AB, Canada, applying to many fruitful Machine Learning courses – Deep Learning and computer programming.