STM JOURNALS

Journal of Computer Technology & Applications
ISSN: 2229-6964 (Online), ISSN: 2347-7229 (Print)
Volume 9, Issue 2
www.stmjournals.com

Temporal Information Extraction from Textual Data using Long Short-term Memory Recurrent Neural Network

Tanvir Hossain, Md. Mostafijur Rahman, S.M. Mohidul Islam*
Computer Science and Engineering Discipline, Khulna University, Khulna, Bangladesh

Abstract

Temporal information extraction from raw text is always challenging. It is time-consuming and sometimes difficult to extract temporal expression manually. For this reason, an automatic system is a demand to find the temporal expressions from the textual data automatically. In this study, we have developed a temporal information extraction system using long short-term memory (LSTM) recurrent neural network (RNN) along with word embedding, where temporal expressions are extracted from TempEval-2 dataset. Performance of the proposed LSTM RNN-based system is highly comparable with the other entries of TempEval-2 challenge. As LSTM RNN can handle both long and short-term dependencies, the proposed system shows a robust result than other renowned existing systems.

Keywords: *LSTM RNN, TempEval-2, temporal information, TIMEX3, word embedding*

****Author for Correspondence*** E-mail: mohid@cse.ku.ac.bd

INTRODUCTION

As the textual data is increasing day by day, the importance of automatically extracting useful information from these vast corpora of natural language is increasing rapidly. It is very difficult to seek particular information manually from a large amount of textual data, because some data may be overlooked as well as it is time-consuming. Temporal information extraction refers to finding out temporal expressions, events, temporal relations, classifying words, and phrases from a large repository of text. Nowadays, it is being used in different fields such as business field, historical data analysis, clinical domain, and much more to get the better outcome [1].

TempEval-2 is included in different evaluation task proposed in SemEval-2007 [2]. The automatic identification of all temporal referring expressions, events and temporal relations within a text is the ultimate aim of research in this area [2]. TempEval-2 is the second edition of TempEval, which is more expanded than the previous edition. The tasks are set in six different languages (Chinese, English, French, Italian, Korean, and Spanish). Moreover, the number of subtasks is six whereas in the last edition it was three.

The intention of the proposed system is to solve the first task of TempEval-2, to determine the extent of temporal expressions from a text corpus. For example, "Football world cup is held in every (four years); next world cup will be organized by Russia. The event starts from (14 June 2018) and will be continuing for (one month). The first match will start at BD time (9:00 pm)." Here, tokens surrounded by parenthesis are temporal expressions. In this research, with the application of long short-term memory (LSTM) recurrent neural network (RNN)-based system, temporal expressions are detected as well as annotated by TIMEX3 tag. For instance, from that quoted text "four years," "14 June 2018," "one month," and "9:00 pm" could have been detected and annotated.

The accomplishment of the task is divided into two steps. The first step is the recognition of temporal expression. To accomplish this step, other systems used different types of machine learning algorithms, such as Markov logic network, conditional random field (CRF), and support vector machine, along with a set of rule-based methods. Word embedding [3] is one of the special techniques to represent same

words by similar value distribution. Word embedding is applied in order to generate patterns by capturing semantic similarities and semantic, syntactic relations of tokens that have been labeled sequentially for the recognition of temporal expressions extent. Then extent of these expressions has been classified to detect them as temporal or non-temporal. In the next stage, handcrafted rules have been applied to merge proper temporal extent. Thereafter, those extents have been annotated by TIMEX3 tag.

The rest contents of this study are organized as follows: Section 2 mentions some related works about temporal expression extraction. In Section 3, we have discussed the stepwise operation of LSTM and application of LSTM RNN over the TempEval-2 dataset. Experimental results obtained by evaluation are inspected in Section 4. Finally, we have concluded in Section 5.

RELATED WORKS

Recently, many approaches have been introduced for temporal information extraction. Strötgen et al. have implemented HidelTime [4] as a UIMA (Unstructured Information Management Architecture) to combine the system with existing document processing pipeline. HidelTime is the winning entry of the TempEval-2.

Derczynski et al. [5] use several natural language processing (NLP) components along with HidelTime tagger within the GATE infrastructure. With their system, temporal expressions have been extracted and normalized in multiple languages and across different domains. Though it has not performed well for any metrics, in future, it may be significant. Another well-renowned system is SUTime [6]; this system has used a cascade of finite automata with heuristics rules. It is also a rule-based system.

UzZaman et al. [7] applied "Trips and Trios" system using CRF and Markov logic network. This system performs comparatively better. Llorens et al. [8] proposed a system using CRF and semantic role labeling. They achieved higher precision of 0.92 and F_1-score of 0.85.

But there are some difficulties in these existing systems. Some of the existing systems cannot resolve long-term dependent expressions and their support for temporal ranges is poor. Usually, machine learning-related methods have to do lots of data preprocessing tasks for which it takes much time for evaluation. Sometimes proper temporal extents are overlooked.

Recently, in the field of NLP, deep learning is being applied widely and has shown good performance. Many researchers are applying deep learning in Twitter data analysis, sentiment analysis, and other related tasks. Tang et al. [9] have proposed four-layered feed forward neural network for differentiating words with opposite sentiment polarities, such as "good" and "bad." They implemented this for classifying Twitter texts and improved the performance. Socher et al. [10] introduced recursive neural tensor networks and analyzed their model on Stanford Sentiment Treebank. Fine-grained sentiment has been classified into five different classes (very negative, negative, neutral, positive, and very positive) in their research. They have achieved the state of the art performance with accuracy of 80.7% and 85.4% for negative/positive classification within a single sentence. Kim et al. [11] have applied Elman and Jordan type RNNs for temporal information extraction. Both approaches achieved comparable results with F_1-score greater than 0.84 and attained precision 0.90 and 0.86, respectively.

PROPOSED METHOD

We have used the TempEval-2 dataset. The dataset is organized in such a pattern that every line consists of one token and its corresponding information, which includes indexes and corresponding labels. Figure 1 shows the system architecture of the proposed method.

According to the frequency of each token, we made a dictionary. The tokens that appear as delimiter are defined as unknown words (UNK). Then the indexing is done according to the most frequent appearance of the words. The most frequent token gets the first index and so on. Table 1 presents words as key and their corresponding frequency as value. The

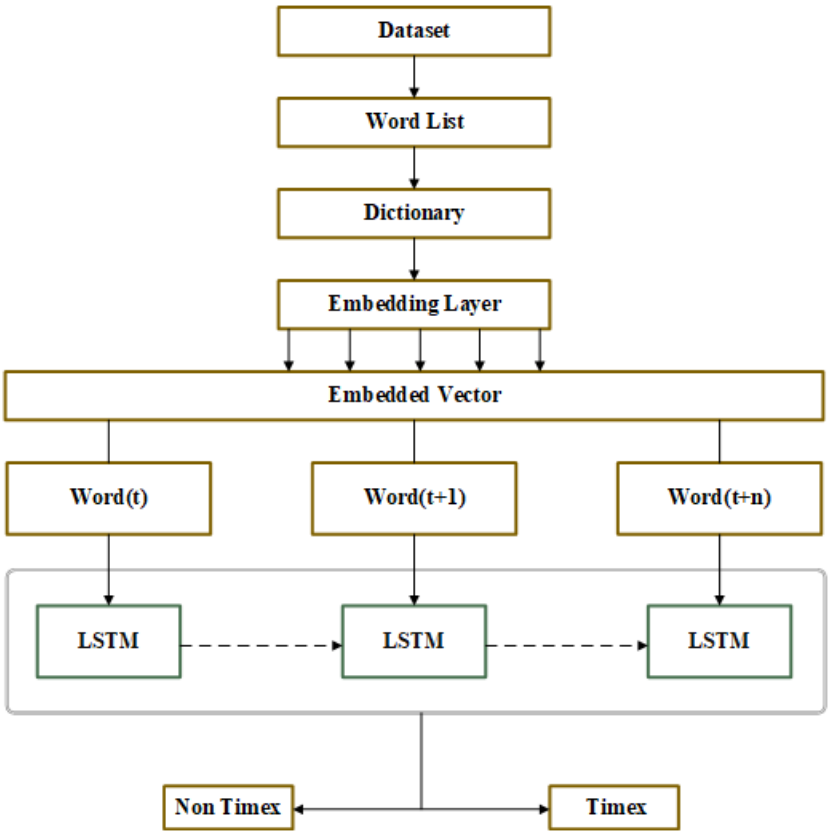

Fig. 1: *Proposed System Architecture.*

indexed words are arranged in a list. This list is the main input of our model. According to this list value, the words are passed through the embedding layer.

Table 1: *Dictionary Table.*

Key (word)	Frequency count
UNK	6,035
The	2,283
Month	30
To	1,287
Friday	49
Million	528
Year	173
In	860
Yesterday	41

Word embedding is the process of representing words as a vector. It is a two-layer network process. It takes text corpus as input and produces a set of vectors as output. We have used 64 dimensions for each vector in the embedding layer. Table 2 presents the words as vector and value of each dimension of each vector.

Table 2: *Embedded Word Vector.*

Word	Value	0	1	…	61	62	63
"In"	860	0.5	0.3	…	0.1	0.03	0.4
"Friday"	49	0.09	0.4	…	0.2	0.1	0.02
"Year"	173	0.2	0.4	…	0.01	0.2	0.05
"Month"	30	0.05	0.02	…	0.15	0.4	0.6
"The"	2,283	0.6	0.04	…	0.08	0.03	0.1
"Million"	528	0.06	0.4	…	0.04	0.30	0.01
"To"	1,287	0.05	0.04	…	0.25	0.05	0.2

The output of embedding layer enters into LSTM RNN as input. The value of each word vector is assigned randomly, where the range of each value is from 0 to 1.

An RNN is a chain of the artificial neural network, where the output of one network is used as input for the next network. Unlike common artificial neural networks, RNNs can use their internal memory to keep information that is used for processing sequence of inputs. Every unit of an RNN has its own output. This output refers to the state of the RNN at a particular time. The main portion of the LSTM network is the cell state, which remains

through the entire network. There are three gates, which are content of the cell state changes: forget gate, input gate, and output gate [12]. Some point-wise operations (element-wise addition, multiplication, and filtering) take place at each layer corresponding to the gate. With these operations, gates decide which information of the cell state will be updated and filtered for the next calculation. Figure 2 shows the basic structure of LSTM RNN [12].

A cell memory or state is a kind of conveyor belt, which runs straight down the entire network. Information just flows along with it. Besides, cell state is the key part of the LSTM network to keep the long-term information. Moreover, cell state is updated for every LSTM cell, so each cell has its corresponding state. Furthermore, cell state value represents the state of each LSTM cell. This cell memory operates as an encoder for corresponding output.

Figure 1 shows that the LSTM cell consists of three sigmoid layers and a *tanh* layer. Each layer has its corresponding function. In the following, the values of forget gate, input gate, and output gate are represented by f_t, i_t, and o_t, accordingly, and the final output of LSTM cell is represented by h_t. Symbol W represents weights between two units of two different layers, such as W_f represents weights for forget gate layer operations. Bias units are represented by b, such as b_i is the bias for input gate layer, h_{t-1} is the output of previous LSTM cell, and x_t is the input of current LSTM cell.

By using the forget gate layer, the LSTM cell decides what information it throws away from the cell state. It looks at previous network's output h_{t-1} and input x_t and outputs a number between 0 and 1. "0" represents "get rid of the whole information" and "1" represents "completely keep the information." After completion of linear algebraic calculations, the calculated value of this gate passes through the sigmoid (σ) layer. Thus, the value of forget gate layer would be occupied.

$$f_t = \sigma(W_f \bullet [h_{t-1}, x_t] + b_f) \quad (1)$$

By operating input gate layer, it decides which value of the cell state should be updated, and next, a *tanh* layer creates a vector for new candidate values C', which is added to the cell state. In the next step, these two are combined for updating cell state information. Formulae for updating cell state are as follows:

$$i_t = \sigma(W_i \bullet [h_{t-1}, x_t] + b_i) \quad (2)$$

$$C'_t = \tanh(W_c \bullet [h_{t-1}, x_t] + b_c) \quad (3)$$

Thereafter, for updating the previous cell state C_{t-1} to C_t, old state is multiplied by f_t to forget or keep the previous information, and then we add multiplicative term $i_t \times C'_t$, which is the new candidate value scaled by how much we decided to update each state value.

$$C_t = f_t * C_{t-1} + i_t * C'_t \quad (4)$$

Next, it will be decided which is going to be the output. This output will be based on the cell state, which is filtered through a *tanh* (to push the values to be between −1 and 1) operational function. Finally, another sigmoid

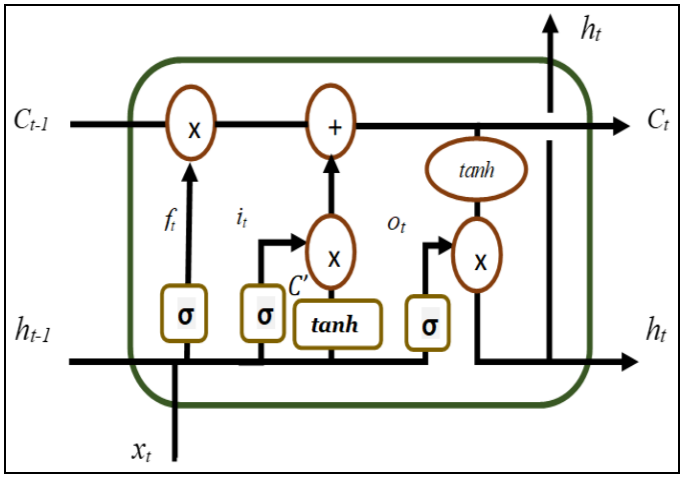

Fig. 2: *LSTM RNN Structure.*

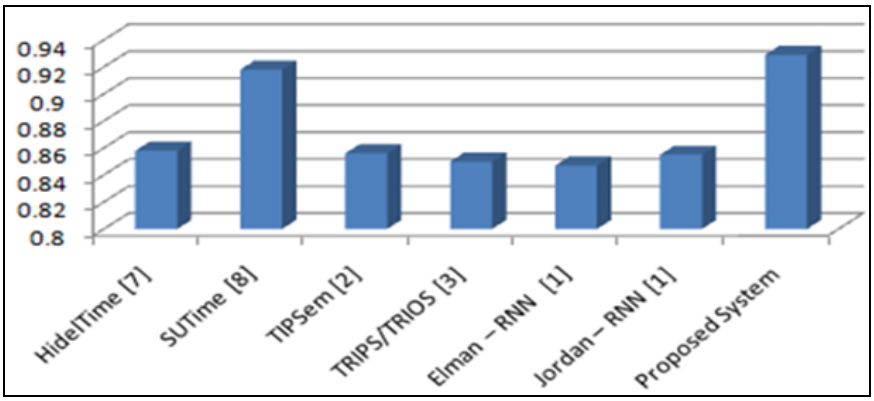

Fig. 3: *F_1-score Value Comparison.*

layer, which decides what parts of the cell state are going to use as output, is multiplied by the filtered value of the cell state, which is determined the final cell output h_t. Result of the output gate layer is calculated as follows:

$$o_t = \sigma(W_o \bullet [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * \tanh(C_t) \quad (6)$$

The output of the LSTM cell at time *t* goes into the next cell as input at time *t* + 1. Simultaneously, LSTM cells have their corresponding input from the embedding layer. For every LSTM cell output, sigmoid activation function is used. According to the output of sigmoid activation function, the type of word is decided whether the word is TIMEX3 or non-TIMEX3.

EXPERIMENTAL RESULTS

We have used batch size 32 for experiment. We have taken 10 epochs to minimize error and optimize our system. The number of the hidden unit used is 16. The learning rate used is 0.01. Table 3 presents the performance comparison of some existing systems with the proposed system.

Table 3: *Performance Comparison.*

System	Precision	Recall
HidelTime [7]	0.90	0.82
SUTime [8]	0.88	0.96
TIPSem [2]	0.92	0.80
TRIPS/TRIOS [3]	0.85	0.85
Elman – RNN [1]	0.90	0.80
Jordan – RNN [1]	0.86	0.85
Proposed system	0.96	0.90

The proposed system shows maximum precision score, 0.96. Though SUTime [6] achieved best result in recall, which is 0.96, the proposed system performs better than other systems except this one. Figure 3 shows the F_1-scores of the mentioned methods.

Proposed method, using LSTM RNN, achieved comparable F_1-score, 0.93, which is greater than other systems. Overall performance is highly comparable with all other entries of the TempEval-2 challenge.

CONCLUSION

Temporal information extraction is a very challenging task in the field of NLP. Here, we have proposed a system for automatically extracting temporal expression from natural language text. The proposed system shows robust result especially for large data, as LSTM can handle both short and long-term dependencies and can extract temporal and non-temporal expression from a text corpus. Our future plan is to classify temporal entities and normalize them.

REFERENCES

1. Lin YK, Chen H, Brown RA. MedTime: A temporal information extraction system for clinical narratives. *J Biomed Inform.* 2013; 46 (Suppl): S20–S28p.
2. Pustejovsky J, Verhagen M. SemEval-2010 task 13: Evaluating events, time expressions, and temporal relations (TempEval-2). *International Workshop on Semantic Evaluations: Recent Achievements and Future Directions;* 2009 June; Boulder, Colorado. pp. 112–116.

3. Mesnil G, Dauphin Y, Yao K, et al. Using recurrent neural networks for slot filling in spoken language understanding. *ASL*. 2015; 23 (3): 530–539p.
4. Strötgen J, Gertz M. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. *5th International Workshop on Semantic Evaluation;* 2010 Jul 15-16; Uppsala, Sweden. pp. 321–324.
5. Derczynski L, Strötgen J, Maynard D, et al. GATE-Time: Extraction of temporal expressions and events. *10th Language Resources and Evaluation Conference*. 2016: pp. 3702–3708.
6. Chang AX, Manning CD. SUTime: A library for recognizing and normalizing time expressions. *Proceedings of the 8th International Conference on Language Resources and Evaluation;* 2012 May 23-25; Istanbul, Turkey. European Language Resources Association: pp. 3735–3740.
7. UzZaman N, Allen JF. TRIPS and TRIOS system for TempEval-2: Extracting temporal information from text. *5th International Workshop on Semantic Evaluation;* 2010 Jul 15-16; Uppsala, Sweden. pp. 276–283.
8. Llorens H, Saquete E, Navarro B. TIPSem (English and Spanish): Evaluating CRFs and semantic roles in TempEval-2. *5th International Workshop on Semantic Evaluation*; 2010 Jul 15-16; Uppsala, Sweden. pp. 284–291.
9. Tang D, Wei F, Yang N, et al. Learning sentiment-specific word embedding for Twitter sentiment classification. *Proceedings of the 52nd Annual Meeting of the Association for Computatioal Linguistics;* 2014 Jun 23-25; Baltimore, Maryland, USA. pp. 1555–1565.
10. Socher R, Perelygin A, Wu JY, et al. Recursive deep models for semantic compositionality over a sentiment Treebank. *Proceedings of the Conference on Empirical Methods in Natural Language Processing;* 2013 Oct 18-21; Seattle, Washington, USA. pp. 1631–1642.
11. Kim ZM, Jeong YS. TIMEX3 and event extraction using recurrent neural networks. *International Conference on Big Data and Smart Computing*; 2016 Jan 18-20; Hong Kong, China. pp. 450–453.
12. Hochreiter S, Schmidhuber J. Long short-term memory. *Neural Comput.* 1997; 9 (8): 1735–1780p.

Cite this Article

Tanvir Hossain, Md. Mostafijur Rahman, S.M. Mohidul Islam. Temporal Information Extraction from Textual Data using Long Short-term Memory Recurrent Neural Network. *Journal of Computer Technology & Applications*. 2018; 9(2): 1–6p.