OPEN ACCESS
Indo-JC
Indonesia Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 5, Issue. 2, September 2020. pp. 91-98
doi:10.21108/indojc.2020.5.2.492

Sequence Chunking on Quran in English Translation using Bidirectional Long Short-Term Memory

Try Arie Rahmat Insani [#1], Moch. Arif Bijaksana [#2]

[#] *School of Computing, Telkom University*
Bandung, Indonesia

[1] tryarie@students.telkomuniversity.ac.id

[2] arifbijaksana@telkomuniversity.ac.id

Abstract

Every Moslem is obliged to read and understand the meanings of the Quran. The problem is the amount of information contained in the Quran so that ordinary people have difficulty understanding the Quran as a whole. Neural networks can be used to extract important information in the Quran to solve this problem. Therefore, the author proposes a model to identify and classify tags using sequence chunking. The system will use the Bi-LSTM model where the system will be given various token from the Quran as the inputs to be identified as the correct tags. The author is using the dataset obtained from website quran.com. The evaluation of the proposed model produces an f-measure value of 0.903.

Keywords: Quran, Sequence Chunking, Bi-LSTM

Abstrak

Setiap Muslim wajib membaca dan memahami makna Quran. Masalahnya adalah jumlah informasi yang terkandung dalam Quran sehingga orang-orang awam mengalami kesulitan memahami Quran secara keseluruhan. *Neural networks* dapat digunakan untuk mengekstrak informasi penting dalam Quran untuk memecahkan masalah ini. Oleh karena itu, penulis mengusulkan model untuk mengidentifikasi dan mengklasifikasikan tag menggunakan *se- quence chunking*. Sistem akan menggunakan model *Bi-LSTM* dimana sistem sistem akan diberikan berbagai token dari Quran sebagai input untuk diidentifikasi sebagai tag yang benar. Penulis menggunakan dataset yang diperoleh dari situs web quran.com. Evaluasi model yang diusulkan menghasilkan nilai *f-measure* sebesar 0,903.

Kata Kunci: Quran, Sequence Chunking, Bi-LSTM

I. Introduction

QURAN is the Moslem holy book that was revealed by Allah SWT to the Prophet Muhammad SAW. Every Moslem is obliged to read and understand the meanings of the Quran and apply it in life to be on the straight path. The Quran contains many historical stories, metaphors, and implied descriptions of the universe and its contents. Because of that knowledge, the Quran is much researched by everyone. The problem is that there is a lot of information contained in the Quran so that ordinary people have difficulty understanding the Quran in its entirety.

To overcome this problem, the author proposes a sequence chunking method to extract and provide important information about the Quran. Sequence chunking is a method to chunk sentences by gathering

Received on Jan 2020. Accepted on Aug 2020

words that are considered important using neural networks. This group of words then become a phrase. This method is divided into two tasks: segmentation to identify the scope of the chunk explicitly and labeling to label each chunk as a unit based on the results of segmentation.

In this paper, the author will implement the Bidirectional Long Short-Term Memory (Bi-LSTM) model to chunk inside-outside-beginning (IOB) tags from the Quran [1]. This model will transform sequence chunking into a sequence labeling problem. The Bi-LSTM will be used to do both segmentation and labeling to correctly identified the tags. The predicted results will be then evaluated with the real tags to calculate system performance. The dataset used by the author is surah 1 (Al-Fatihah) to surah 5 (Al-Maidah) from Saheeh International Quran obtained from *quran.com*. The author chose the Saheeh International because it provides convenience to the reader and it brings closer to original meanings [2]. The dataset is still not in the appropriate form, therefore, some preprocessing needs to be done.

II. Literature Review

A. Sequence Chunking

Sequence chunking is a deep learning model for text chunking. This model is created to overcome two drawbacks of the IOB scheme to label chunks. First, no explicit model to learn and identify the scope of chunks in a sentence, and second, some neural networks have the ability to encode context information but don't treat each chunk as a complete unit [1].

In sequence chunking, the task is divided into two sub-tasks. First is segmentation, to identify the scope of the chunks explicitly and second is labeling, to label each chunk as a single unit based on the segmentation results.

There have been some similar research to implement sequence chunking in text mining. Some of its application are named entity recognition (NER) using character-level sequence-to-sequence model and understanding dialogue using sequence-to-sequence based on semantic [3] [4].

B. Recurrent Neural Network

Recurrent Neural Network (RNN) is one of the types of artificial neural network based on David Rumelhart's work in 1986 [5]. It is used for modeling sequential information because it can recognize sequential characteristics and predict the next scenario. It can be used for keyword spotting, handwriting recognition, and speech recognition [6] [7] [8].

Even though theoretically it can still capture long-distance dependencies, in practice RNN still suffers from gradient vanishing [9]. To solve this problem, Long Short-Term Memory was introduced. Long Short-Term Memory (LSTM) is an RNN architecture that is used in deep learning and was first introduced by Hochreiter & Schmidhuber in 1997 [10].

The Bi-LSTM is a modification of the LSTM and based on bidirectional recurrent neural networks (BRNN) [11]. It duplicates the LSTM layer into two layers: forward LSTM layer, that reads the inputs as it is and computes the forward hidden states and backward LSTM layer, that reads the inputs in the reverse order and computes backward hidden states. The hidden states of the Bi-LSTM are produced by concatenating both hidden states. Bidirectional networks outperform unidirectional networks because it analyze both directions [12].

C. IOB Tags

IOB tags are a tagging format used in computational linguistics. This format was first introduced by Ramshaw and Marcus in their paper entitled "Text Chunking using Transformation-Based Learning" in 1995 [13]. Prefix B- is used for the beginning of a chunk while prefix I- is used for the inside of a chunk. The O

tag indicates that the word token is outside the chunk.

D. Precision

Precision is the correct percentage of phrases detected [14]. To achieve high precision values, words that are correct by the system, but are false manually (false positive) must be removed. The precision is calculated using the formula (1) [15].

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (1)$$

E. Recall

Recall is the percentage of completeness of the founded phrase [14]. To achieve a high recall value, the wrong founded words by the system, but manually correct (false negatives) must be eliminated. The recall is calculated using the formula (2) [15].

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (2)$$

F. F-Measure

F-Measure is the average harmonic of precision (P) and recall (R) [14] [16]. The f-measure is calculated using the formula (3) [15].

$$F\_Measure = \frac{2 \cdot P \cdot R}{P+R} \quad (3)$$

III. RESEARCH METHOD

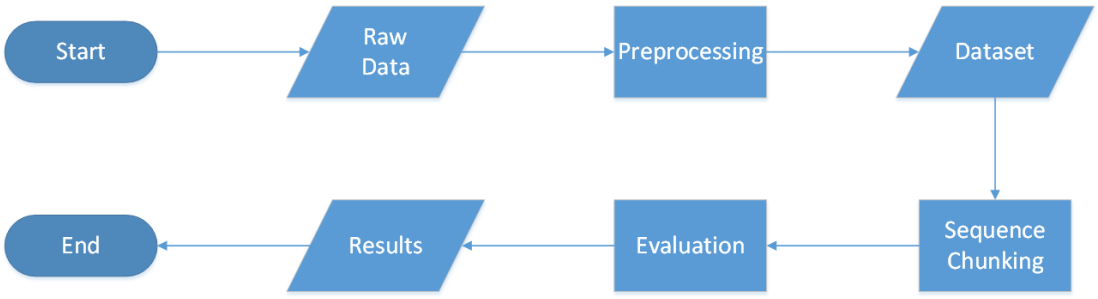

Figure 1: General overview of the system

Figure 1 explains the flowchart of the system. The system will preprocess the raw data first so it can be read. The preprocessed dataset then will be fed into sequence chunking model. The result of sequence chunking will be evaluated and then outputted the evaluation result.

The model that will be used in this research is the Bi-LSTM model. It uses Bi-LSTM, a modification of the LSTM, for both segmentation and labeling of the Quran. It will transform sequence chunking into a sequence labeling problem. Unlike other proposed models, Bi-LSTM model is easy to implement and understand which is the reason why it is chosen for this research. The model can be seen in Figure 2.

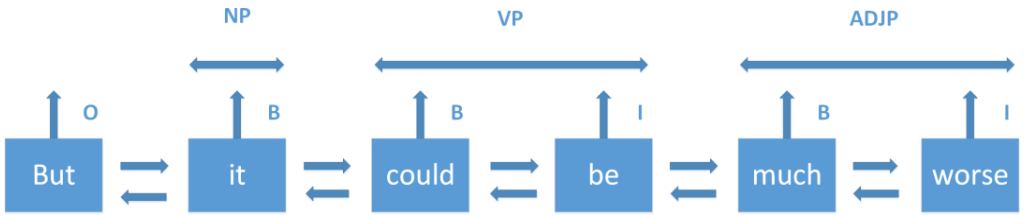

Figure 2: Bi-LSTM Model

A. Dataset

The dataset used for research is Quran verses from surah 1 (Al-Fatihah) to surah 5 (Al-Maidah). They are obtained from *quran.com* in *text* file format with a total of 789 verses. However, the data is not in an appropriate form. Therefore, some preprocessing needs to be done. The obtained data will be converted into a *tsv* (Tab-separated value) format to make preprocessing more convenient. After preprocessing, the author will manually apply the token with a tag. After the applying is done, the dataset is ready to use. The dataset will be divided into two. 80% data as training data and 20% data as test data.

B. Preprocessing

Preprocessing methods are chosen based on the form of data obtained from the source. Several preprocessing methods that are used in the system, as follows:

1) Brackets Removal: The raw data still has brackets "[]" from the sentence, therefore it must be removed. The example can be seen on Table I.

Table I: Example of Brackets Removal

Input	Output
[All] praise is [due] to Allah , Lord of the worlds -	All praise is due to Allah , Lord of the worlds -

2) Tokenization: After removing brackets, each word in the sentence will be broken down into token. The example can be seen on Table II.

Table II: Example of Tokenization

Input	Output
All praise is due to Allah , Lord of the worlds -	"All", "praise", "is", "due", "to", "Allah", ",", "Lord", "of", "the", "worlds", "-"

3) Labeling: After breaking down each word, the word tokens will be assigned a tag using IOB format. The tag will be "B-NP", "I-NP", and "O". The process of labeling is done using *nltk* module. The example can be seen on Table III.

Table III: Example of Labeling

Word	Tag
All	B-NP
praise	I-NP
is	O
due	O
to	O
Allah	B-NP
,	O
Lord	B-NP
of	O
the	B-NP
Worlds	I-NP
-	O

4) Max Length: After assigning tag, the author determines the max length using visualization of distribution max length per sentence. From the figure 3, it shows that the length is mostly distributed on 60-80 words. Therefore, 80 is chosen as the max length.

5) Token Indexing: Each token of both word and tag will be indexed by number to create a sequence. To turn them into numeric, the researcher creates a vocabulary. The vocabulary contains all token sorted alphabetically with an index number. The system will convert a token into numeric based on the vocabulary. The example can be seen on Table IV.

Table IV: Example of Token Indexing

Word	Index
PAD	0
Arafat	4
Be	5
Believe	6
Imran	7

6) Padding: The model only accepts a fixed-length sequence, so each sentence will be padded to the chosen max length. The researcher uses *keras* module, namely *pad_sequences*, to pad the sequence. The example can be seen on Table V.

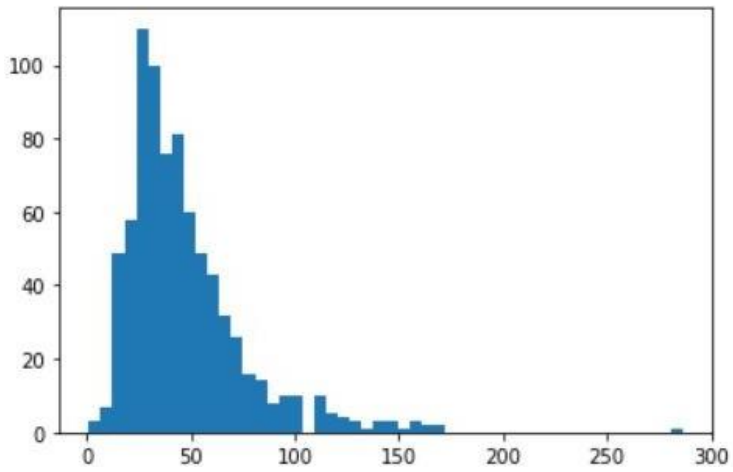

Figure 3: Distribution of max length per sentence

Table V: Example of Padding

Input	Output
1168, 710, 2292, 1054, 930, 1477, 710, 135, 1297, 1477, 710, 344, 1297, 634	1168, 710, 2292, 1054, 930, 1477, 710, 135, 1297, 1477, 710, 344, 1297, 634, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

7) Split Data: Lastly, the data will be split into training and test data. The researcher uses *sklearn* module, namely *train_test_split*, to split the data. Training data is 80% of the whole data and the rest is for test data. Both training data and test data contain words and tags. The size of training data is 631 while the size of test data is 158.

C. Sequence Chunking

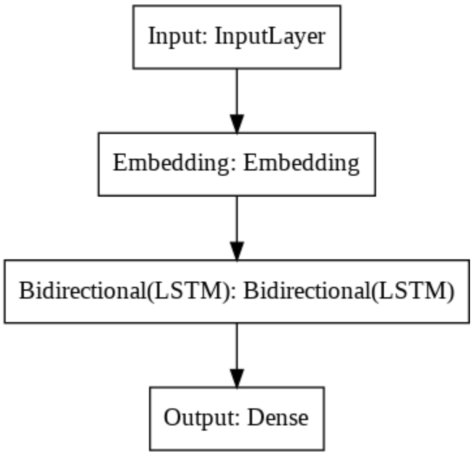

Figure 4: Bi-LSTM Plot

The plot of the model can be seen in Figure 4. In training the proposed model, the author uses *adam* as an optimizer function and *categorical crossentropy* as the loss function. The author also uses batch size of 64, epochs of 200, and validation split of 0.1. The model will fit both words and tags of training data for training. The process of the model as follows:

1) Input: First, the data is inputted into the input layer where the data is instantiated as a Keras tensor.

2) Embedding: After the data is instantiated as a Keras tensor, the data is embedded in the embedding layer where it turns positive integers of inputs into dense vectors of fixed size.

3) Bi-LSTM: The embedded data is then inputted into an LSTM layer wrapped in the bidirectional wrapper. The author uses RNN units of 100 and a dropout rate of 0.5, following the values in the paper "Neural Models for Sequence Chunking". The outputs are sequences of a tensor.

4) Dense: Lastly, the outputs of Bi-LSTM will be inputted into Dense layer with softmax activation.

D. Evaluation

After the system predicting the tags of the inputs, it will calculate the performance of the system and each tag. The system uses Precision, Recall, and F-Measure to measure system performance.

IV. RESULTS AND DISCUSSION

Table VI: System Performance Result

Metric	Score
Precision	91.7%
Recall	88.9%
F1-Measure	90.3%

Table VII: Classification Performance Result

	Precision	Recall	F1-Score	Support
B-NP	0.94	0.91	0.93	1929
I-NP	0.91	0.85	0.88	523
O	0.98	0.99	0.99	10188
Accuracy	-	-	0.97	12640

From Table VI and Table VII, the performances of the system are quite satisfying. The system achieves a high score in all three metrics measurement. Not only that, but the system also has a high score in both predicting IOB tags and accuracy with the values mostly over 90%. These results show that the Bi-LSTM model is a success.

The 'O' tag has the highest value among the tags which shows that the system predicts 'O' tag more correctly. The high precision explains that most of the predicted 'O' tag is correct and the high recall explains that the system finds almost all 'O' tag completely. Because both precision and recall have high value, the f1-measure is also high.

Meanwhile, the 'I-NP' tag has the lowest value among the tags which shows that the system has a hard time predicting' I-NP' correctly. Even though the precision is high, the recall is lower which shows that the system cannot find all tags completely which affects f1-score.

V. CONCLUSION

It is proved that the Bi-LSTM model for sequence chunking has a promising performance in identifying tags in the Quran. The system achieves a high score in all three metrics measurement. Not only that, but the system also has a high score in both predicting IOB tags and accuracy with the values mostly over 90%. In the future, this model can be used as a benchmark for other sequence chunking models for Quran in English Translation.

ACKNOWLEDGMENT

The author would like to thank Human Language Technology Laboratory for their assistance in teaching deep learning and suggesting several modifications for the proposed system. Without their guidance and support, the author won't be able to finish this research.

REFERENCES

[1] Feifei Zhai, Saloni Potdar, Bing Xiang, and Bowen Zhou. Neural models for sequence chunking. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[2] Saheeh International. *The Qur'an: English Meanings*. Abul-Qasim Publishing House, 1997.

[3] Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. Sequence-to-sequence data augmentation for dialogue language understanding. *arXiv preprint arXiv:1807.01554*, 2018.

[4] Yanyao Shen, Hyokun Yun, Zachary C Lipton, Yakov Kronrod, and Animashree Anandkumar. Deep active learning for named entity recognition. *arXiv preprint arXiv:1707.05928*, 2017.
[5] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
[6] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. In *International Conference on Artificial Neural Networks*, pages 220–229. Springer, 2007.
[7] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.
[8] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.
[9] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.
[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
[11] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.
[12] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610, 2005.
[13] Lance A Ramshaw and Mitchell P Marcus. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer, 1999.
[14] Tong Zhang, Fred Damerau, and David Johnson. Text chunking using regularized winnow. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 539–546, Stroudsburg, PA, USA, 2001. Association for Computational Linguistics.
[15] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.
[16] Yutaka Sasaki et al. The truth of the f-measure. *Teach Tutor mater*, 1(5):1–5, 2007.