2023 24th International Carpathian Control Conference (ICCC) | 979-8-3503-1022-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCC57093.2023.10178946

Automatic Question Generation based on Sentence Structure Analysis

Walelign Tewabe Sewunetie
Institute of Informatics
University of Miskolc
Miskolc-Egyetemvaros, Hungary
walelígntewabe@gmail.com

László Kovács
Institute of Informatics
University of Miskolc
Miskolc-Egyetemvaros, Hungary
kovacs@iit.uni-miskolc.com

***Abstract*—Automated question generation is a key problem in computational linguistics and knowledge engineering. There are different approaches and methodologies like template-based, syntax-based, or machine learning approaches. The paper investigates the efficiency and analysis of the role of the embedding module in sentence parsing. Embedding language models are important modules and the performed tests show that embeddings can provide a high accuracy in semantic-level similarity methods.**

***Index Terms*—Computational Linguistics, Automated Question Generation, Neural Network, Semantic Similarity**

I. Introduction

Online learning is becoming more common and it allows students to access online materials anywhere at any time. In this information era, many organizations and institutions provide a variety of training alternatives to their employees or learners [1]. Due to the radical expansion of the internet over the past two decades, more individuals have got access to online resources [2]. As a result of this development, e-learning is quickly gain popularity as a teaching method, particularly in higher education. The assessment phase, which is used to gauge the academic success of the students, is one of the key difficulties in e-learning. The process of automatically creating questions from different inputs, such as raw text, databases, semantic representations, ontologies, taxonomies, knowledge bases, images, or audio and videos known as Automatic Question Generation (AQG) [3].

Researchers [4], from various fields have recently demonstrated a common interest in using AQG for educational reasons. An important function in educational assessment played by AQG is to generate questions and their answers [5]. According to the survey [6], the main challenges in AQG development are the following issues: question generation from multiple sentences, short and long-type answer assessment, question generation, and assessment using machine learning tools. In this paper, we focus on the presentation of a neural network-based approach.

The rest of the paper is organized as follows: In the second section, we discussed the state-of-the-art of the most recently developed related works focusing on template-based AQG technique. Thirdly, we discussed the main concepts of embedding modules.

In section four, we presented the concept of query generation using a neural network with embedding. Section five, presented the proposed query generation using a similarity approach with an embedding approach. In the sixth, the experimental result and evaluation are presented, and finally, the conclusion and future works are discussed.

II. Template-based AQG

AQG is defined as the process of producing relevant, accurate, and syntactically sound [7] questions from various input formats, and it is the current challenging research in the field of natural language processing (NLP). Among various AQG methods [8], a template-based method is the oldest, and it can be easily implemented. Template-based method use templates containing fixed text, and some placeholders that are populated from the given content. According to the thesis report [9], the templates are created by focusing on the events (actions, happenings), and existents (characters, settings).

In addition, we observed that most current templates ask about the subject, the predicate, and the object of the events and existents [10]. In the development of the template-based question generation method, first, we need to prepare a quality and representative dataset. While, due to the complexity of natural language structure, it is very difficult to create general-purpose templates. [11]. In this study, we have developed develop an open-ended template-based system.

After analyzing the structure of many sentence question pairs collected from various domains, we developed a template by considering most of the wh-question words (what, how, who, how many, where do, which, what kinds, when, why). Finally, we developed 48 open domain rule sets, according to this we have also created our own syntax based-template. For testing, we have prepared our own 15-sentence-question pair test dataset collected from various sources. We used this test dataset for all systems that we analyzed for this study. From our test result, we observed that most generated questions have naturalness and meaningfulness problems. For this reason, we focused to evaluate by using naturalness and meaningfulness as basic parameters.

In the implementation of AQG, we used spaCy's improved Named Entity Recognition (NER) [12], which, how, how much, how many, which is capable of labeling more entity types, including money, dates/times, e.t.c, and it is important for selecting a proper wh-question word i.e what, who, where, when, [4]. Even if NER can identify the named entities automatically, [13] it is incapable to identify all categories of 'person', 'organization', 'location', and so on.

III. Embedding modules

The first thing that happens when a piece of text is passed through a network is to transform the words into a representation that the network can manipulate [14]. The words can be transformed into real-valued vectors of a specific constant dimensional space to do this. The term "word embedding" refers to the vector representation of words. These word embedding's can be randomly initialized while training a neural network (NN), and their weights can be adjusted along with the network's other trainable parameters [15].

The most common resources are trained based on the following techniques with a considerable amount of data. Word2vec [16] is one of the word embedding techniques that use a statistical method to extract word embedding from a given text corpus. To get context based on the words present in a particular window, it may be based on the Continuous Bag of Words (CBOW) [17]. The Skip-Gram Model [18], on the other hand, may also be used in this situation to forecast a word's surroundings. This approach has previously produced pre-trained word embedding that is based on the Google News corpus, which contains over 100 billion words. Vectors with 300 features or other values are used to represent these words.

IV. Query Generation using Neural Network with Embeddings

In this approach, a neural network classifier is used to predict the query sentence. The proposed model uses an MLP neural network model, where the MLP network is able to predict a single value. In order to predict a word sequence, the system contains more neural networks, each model is responsible for a single word position. If M denotes the maximal length of the investigated sentences, the architecture involves M MLP modules and the output sentence is constructed as the sequence of the generated outputs.

Considering the MLP units, each neural network unit has the same input, namely the feature vector of the corresponding sentence. The content of the feature vector involves the following elements for every word in the input sentence: POS category, NER category, Morphological units, and Word context description.

For the description of the word context, we use the Word Embedding model. The embedding model is used mainly in natural language processing, where the model generates a high dimensional vector for every word including its context. In the generated vector space of the embeddings, words with similar meanings (like dog and cat) have positions near to each other.

The formal model of the proposed algorithm is based on the following elements.

1) training set $T : \{t_i\}$ where t_i is equal to a triplet (s_i, p_i, q_i). The first component is the input sentence; the second component is the position of the target word and the third tag is equal to the generated query sentence.
2) The input and output sentences are transformed into a lemmatized form in order to increase the unambiguity of the sentence form.
$$s'_i = lemma(s_i), q'_i = lemma(q_i). \quad (1)$$
In this preprocessing step, every variant of the same word is mapped to the same word format.
3) In the case of a query sentence, we perform a simplification phase to reduce the set of possible target words. This step is based on the consideration that most words in the query are related to some word in the input sentence. In the preprocessing step, every word related to the input sentence is converted to a relative format $\_n$ where n denotes the related position in the input sentence.
4) N: the set of MLP neural networks, with $N_i : T \rightarrow W$, where W is the union of two sets: a) the set of relative words from the input sentence $(\_1, \_2, ...)$ b) the set of words not given in the input sentence.
5) The training set $T : \{t_i\}$ is converted into (X, Y), where X denotes the calculated feature set for $\{s_i\}$ and Y is the encoded $\{q_i\}$ set.
6) Split the (X, Y) into (X_i, Y_i) where i denotes a position index and $X_i = X$ for every position.
7) Train the generated N_i MLP networks.

In the case of prediction, the following steps are executed:

1) Input is a pair (s, p) where s is the base sentence and p is the position of the target word.
2) $s' = lemma(s)$
3) For every positions $l \in [0, ..., M]$, we perform a prediction: $c_i = N_i.pred(s'_i)$ the output categories c_i are converted into word lemmas w'_i using the s_i and the additional dictionary.
4) $q'''_i = [w'_0, w'_1, ..., w'_L]$, where w'_{L+1} is the first terminal symbol in the sequence $(L < M)$.
5) Converting q'''_i into the output q''_i using a NLP engine.

For the evaluation of the engine accuracy, we used two measures:

- $acc_1 = Avg(\sum_{l=1}^{M} d_{nl})$, where d_{nl} denotes characteristic variable for the n-th sample at the l-th position. Its value is equal to 1 if the predicted word is equal to the real word at the given position, otherwise, the value is equal to 0.
- $acc_2 = Avg(1 - edit\_distance(q'_i, q''_i)/M)$. In this case, the edit distance value is used to measure the similarity between the predicted and measured values. The main motivation for acc_2 is the fact that acc_1 is too strict, it requires position-level equivalences.

In the test framework, we have used the medium language model of the spaCy Python framework. In this model, the output vector of the embedding module has 300 dimensions.

V. Query Generation using Similarity Ranking with Embeddings

In this approach, we use a direct sample selection to predict the corresponding query sentence. The method selects the sample triplet (s_i, p_i, q_i) with the highest similarity with the query sentence input pair (s_i'', p_i''). The similarity value is measured with a semantic similarity using the language embedding model. The main benefits of this similarity approach are

- context sensitivity
- high-level independence from the surface layer
- simple application.

The formal model of the proposed system:

1) Construction of the sample dictionary from the training set $T : \{t_i\}$ where t_i is equal to a triplet (s_i, p_i, q_i). The first component is the input sentence; the second component is the position of the target word, and the third tag is equal to the generated query sentence
2) The target input is a pair (s'', p'').
3) The input and target sentences are transformed into a lemmatized form in order to increase the unambiguity of the sentence form.

$$s_i' = lemma(s_i), q_i' = lemma(q_i).s''' = lemma(s_i'') \tag{2}$$

In this preprocessing step, every variant of the same word is mapped to the same word format.

4) Calculate the embedding level similarity for the dictionary samples:

$$c_i = embedding\_similarity(s_i', s'') \tag{3}$$

This measure takes the whole sentences into account, but it does not consider which is the target word in the sentences.

5) Sorting the dictionary sample in decreased order of similarity.
6) Process the dictionary samples in this order to measure a finer similarity value considering also the target word selection.
7) We calculate a word-level similarity matching for the components is the investigated pair (s_i', s''), each word is assigned to the nearest partner word using a pairing algorithm.
8) Select the first sentence in this order, where the target words are pairs in the finer level similarity matching.
9) Calculate the output query sentence based on the winner sample using corresponding substitutions.

For the evaluation of the engine accuracy, we use only one measure, the acc_2 accuracy value.

VI. Testing and Evaluation

In the efficiency comparison framework, the following four methods were implemented:

- AQG using sentence matching with syntax-templates (M_ST)
- AQG using similarity ranking with embeddings (M_SE)
- AQG using MLP neural network without embeddings vector (M_NN)
- AQG using MLP neural network with embeddings vector (M_NNE).

The training and test datasets were generated from three sources: synthetic data generation, Manual data generation, and External databases.

In the first generation approach, we have constructed a world domain with a limited number of words and the sentences were constructed with application of sentence templates. Here are some examples of the training items:

```
s: Peter is going to farm afternoon
p: 2
q: What is doing Peter afternoon

s: Anna is travelling to city today
p: 3
q: Where is Anna travelling to today
```

The algorithms were implemented in Python framework using the following standard libraries: NLTK, spaCy, and Keras.

Regarding the implementation of the method using syntax-templates, we have used the template ruleset presented in [7].

The tests were evaluated for the following training set sizes: Small: $N = 80$, Medium: $N = 800$, Large: $N = 8000$

The test results for acc_2 measure are summarized in tables Table.I- Table.III. The cells show the measured accuracy in percentage (%). The presented values are the average values of 6 measures.

TABLE I
Accuracy with small training set

N	E	M_ST	M_SE	M_NN	M_NNE
80	5	46	85	44	43
80	20			42	41
80	100			38	39
80	300			36	37

TABLE II
Accuracy with medium training set

N	E	M_ST	M_SE	M_NN	M_NNE
800	5	47	92	47	48
800	20			48	49
800	100			51	51
800	300			52	51

The test results show some interesting experiences, namely:

TABLE III
ACCURACY WITH LARGE TRAINING SET

N	E	M_ST	M_SE	M_NN	M_NNE
8000	5	47	92	47	48
8000	20			51	51
8000	100			52	51
8000	300			53	52

- There is no significant differences between the tested MLP neural network models with embeddings and without embeddings. This fact shows that the syntax level attributes (like POS value, syntax role) have enough power to achieve a moderate accuracy, the embedding vector will carry information similar to the syntax oriented attributes.
- The embedding module is very effective in the AQG method using similarity matching. This method could achieve the best results, over 90% accuracy. The main benefit of this approach, that it can detect the dictionary items most similar to the input sentence for query generation. In this case, the similarity measure covers also the semantic viewpoint, too.
- The syntax-template approach could provide an accuracy similar to the accuracy value of the neural network approach, especially for smaller data sets and smaller epoch numbers. The main problem of this approach is that it is very rigid and it is very time-consuming to extend the applied ruleset.

VII. Conclusion

The paper presented baseline experiments to compare the efficiency of some key methods for automated question generation. In the method comparisons, the paper focused on the role and efficiency of the language embedding modules. The test results on the comparison of four methods (AQG using sentence matching with syntax templates; AQG using similarity ranking with spacy embeddings; AQG using MLP neural network without embeddings vector; AQG using MLP neural network with spacy embeddings) show that the embedding module is very effective in the method using similarity matching. We can also conclude that a combination of the investigated methods (neural network, syntax similarity, and semantic similarity method) is a very promising approach for future investigations.

Acknowledgements

The research reported here was carried out as part of the EFOP-3.6.1-16-2016-00011 "Younger and Renewing University – Innovative Knowledge City – Institutional development of the University of Miskolc aiming at intelligent specialization" project implemented in the framework of the Széchenyi 2020 program. The realization of this project is supported by the European Union, co-financed by the European Social Fund.

References

[1] N. J. Navimipour and B. Zareie, "A model for assessing the impact of e-learning systems on employees' satisfaction," *Computers in Human Behavior*, vol. 53, pp. 475–485, 2015.
[2] R. Säljö, "Digital tools and challenges to institutional traditions of learning: technologies, social memory and the performative nature of learning," *Journal of computer assisted learning*, vol. 26, no. 1, pp. 53–64, 2010.
[3] L. Pan, W. Lei, T.-S. Chua, and M.-Y. Kan, "Recent advances in neural question generation," *arXiv preprint arXiv:1905.08949*, 2019.
[4] G. Kurdi, J. Leo, B. Parsia, U. Sattler, and S. Al-Emari, "A systematic review of automatic question generation for educational purposes," *International Journal of Artificial Intelligence in Education*, vol. 30, no. 1, pp. 121–204, 2020.
[5] K. László and S. W. Tewabe, "Challenges in target selection for automatic question generation," *Multidiszciplináris Tudományok*, vol. 9, no. 4, pp. 500–507, 2019.
[6] B. Das, M. Majumder, S. Phadikar, and A. A. Sekh, "Automatic question generation and answer assessment: a survey," *Research and Practice in Technology Enhanced Learning*, vol. 16(1), pp. 1–15, 2021.
[7] W. T. Sewunetie and L. Kovács, "Evaluation of python based nlp frameworks for template based automatic question generation," *Production Systems and Information Engineering*, p. 54, 2020.
[8] G. Danon and M. Last, "A syntactic approach to domain-specific automatic question generation," *arXiv preprint arXiv:1712.09827*, 2017.
[9] E. L. Fasya, "Automatic question generation for virtual humans," 2017. [Online]. Available: http://essay.utwente.nl/73352/
[10] M. Heilman, "Automatic factual question generation from text," 2011.
[11] T. Alsubait, B. Parsia, and U. Sattler, "Ontology-based multiple choice question generation," *KI-Künstliche Intelligenz*, vol. 30, no. 2, pp. 183–188, 2016.
[12] K. Mhatre, A. Thube, H. Mahadeshwar, and A. Shrivas, "Question generation using nlp," *International Journal of Scientific Research & Engineering Trends*, vol. 5, no. 2, pp. 2395–566, 2019.
[13] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 50–70, 2020.
[14] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[15] S. Takase and S. Kobayashi, "All word embeddings from one embedding," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3775–3785, 2020.
[16] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.
[17] W. Ling, Y. Tsvetkov, S. Amir, R. Fermandez, C. Dyer, A. W. Black, I. Trancoso, and C.-C. Lin, "Not all contexts are created equal: Better word representations with variable attention," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1367–1372.
[18] C. Xu, Y. Bai, J. Bian, B. Gao, G. Wang, X. Liu, and T.-Y. Liu, "Rc-net: A general framework for incorporating knowledge into word representations," in *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, 2014, pp. 1219–1228.