AUTOMATIC QUESTION-ANSWER PAIRS GENERATION FROM TEXT

Holy Lovenia[1]*, Felix Limanta*[2]*, Agus Gunawan*[3]

Department of Informatics, Bandung Institute of Technology
holy.lovenia@gmail.com[1], felixlimanta@gmail.com[2], agusgun@protonmail.com[3]

ABSTRACT

Possible opportunities for question-answer generation have been suggested in the previous work, including in the field of education. The need of questions and answers is prompted for various purposes, e.g. self-study, academic assessment, and coursework. However, the conventional way to create question-answer pairs has been both tedious and time-consuming. In the present study, we propose an automatic question generation for sentences from text passages in reading comprehension. We introduce a rule-based automatic question generation for the task, as well as implement statistical sentence selection and various configurations of named entity recognition. Three types of -questions ("What", "Who", and "Where") can be produced by our system. The system performs well on generating questions from simple sentences, but falters on more complex sentences due to incomplete transformation rules.

Index Terms— Question-answer generation, named entity recognition, sentence selection, constituent parsing

1. INTRODUCTION

The translation of machine-readable, non-linguistic information into human language representation is the task of natural language generation (NLG), which is one of substudy of natural language processing (NLP). Question-answer generation (GAQ) from text is classified as NLG task focused on generating question-answer pairs from unstructured text. Basically, there are several NLG tasks related to question-answer generation, such as content determination (to decide which information should be involved) and linguistic realization (to apply grammar rules to produce valid sentences) [1]. However, in order to be able to generate questions and answers from text, the system has to understand what it processes first. Then, natural language understanding (NLU) also should be included in question-answer generation to convert human language into representations that the machine understands [2].

It has been suggested that question-answer generation has numerous possible, useful applications. QAG can be used to suggest frequently asked questions (FAQ) related to pre-defined documents or other media, to generate questions needed to assess deeper learning (self-learning help), and to provide question-answer pairs for tutoring purposes. There are also several applications of QAG that lie outside education field, such as to give suggestions about questions that might be asked in legal or security contexts [2].

In the present study, we aim to focus on generating question-answer pairs for academic purposes. This is motivated by the disadvantages of using the conventional way to produce question-answer pairs related to a certain text, which is considered time-consuming and repetitive. Given the need for QAG without human efforts in the academic field, our main objective in the present study is to build an automatic QAG system as an authoring aid to produce answers and the related questions based on a short reading comprehension passage. Moreover, the second objective is to run experiments on each module in the system separately. It is needed to confirm that the system is using the best configuration of each module.

2. RELATED WORK

There are several approaches to accomplish question-answer generation. Du et al. introduced the usage of end-to-end fashion and deep sequence-to-sequence learning to generate questions for reading comprehension [3]. They implemented an attention-based sequence learning with two encodings, namely sentence encoding and paragraph-level information. Their system, Neural Question Generation (NQG), is able to generate "what", "when", "where", "who", "why", and "how" questions (without answers). There is also an ongoing recent work about question generation by Sarvaiya, A., whose framework we adopt and develop into our system [4]. The strategy used in this study is selecting important sentences, finding candidate gaps in sentences, then forming the interrogative sentences.

3. QUESTION-ANSWER GENERATION

In this section, we propose an approach which is generally based on the framework of ongoing work by Sarvaiya in

2018 [4]. Formally, given a passage P, question-answer generation (**QAG**) system retrieves the most important sentence S from P. Then, **QAG** system produces a set of question-answer pairs $\{(Q_j, A_j)\}$, where each generated A_j can be found in S, and its pair Q_j is the interrogative version of S or a clause C_k from a set of clauses $\{C_k\}$ in S, without A_j in it. As shown in Fig 1, here are four main modules in our **QAG** system.

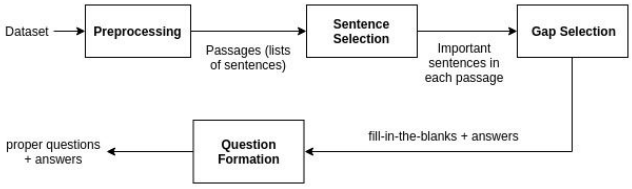

Fig 1. System architecture in overview

1. **Preprocessing**, which cleans the input passage P from unnecessary characters and shapes it into the desirable form (list of sentences).
2. **Sentence Selection**, which picks top-N most important sentences $\{S_1, ..., S_N\}$ given P. The text summarization method used can be chosen between TextRank, multi-word phrase extraction (MWPE), and latent semantic analysis (LSA). The chosen method ranks the sentences in P and selects the top-N highest ranked sentences as the output. In our system, N equals to 1.
3. **Gap Selection**, which selects phrases in S that can be used as answers $\{A_j\}$ based on constituent tree from syntactic parser and named entity recognition (NER).
4. **Question Formation**, which creates the interrogative version of S or $C_k \in \{C_k\}$ in S without A_j to make a question Q_j for each answer in $\{A_j\}$. The final output of this module is question-answer pairs $\{(Q_j, A_j)\}$ related to P.

4. SENTENCE SELECTION

Sentence selection module was implemented using text summarization methods. One of the methods is TextRank [5], which is is a graph-based ranking model for graphs extracted from natural language texts. TextRank makes an assumption that the most important sentences are the ones that are the most similar to every other sentence. The similarity can be computed using cosine distance or Jaccard distance. It utilizes PageRank algorithm by Google Search [6] to rank the importance of each sentence in the passage. The outline of **TextRank** is as follows.

1. Each sentence in passage P is added as vertex in the graph
2. Calculate similarity between every pair of sentences and use it as edge in the graph
3. Normalize the similarity values
4. Run PageRank algorithm until convergence
5. Rank the sentences based on their scores
6. Pick top-N sentences as the summarization

In the second method, multi-word phrase extraction (MWPE), we used TextRank to extract key phrases instead of the highest ranked sentences. Then, the occurrences of words that formed the key phrases were counted in each sentence of the passage. The most important sentences were the ones that had the highest number of occurrences. The following is the outline of **multi-word phrase extraction**.

1. Each tokenized word is annotated with part-of-speech (POS) tags
2. Filter the words (only leave those that are nouns or adjectives)
3. Add the words to the graphs as vertices
4. Add an edge between words that co-occur within window size W words of each other
5. Run PageRank algorithm until convergence
6. Rank the words based on their scores
7. Pick top-K words as keywords
8. Pair the keywords together as key phrase if they are adjacent in the graph
9. Calculate the occurrence of words that form the key phrases in each sentence
10. Pick top-N sentences with the highest number of occurrences as the summarization

Our sentence selection module is also able to choose latent semantic analysis (LSA) as the text summarization method. We use LSA method that was proposed by Steinberger and Ježek [7]. In LSA, it uses the context of the passage and information such as term usages to identify the semantic relation between sentences. It does not use syntactic relation, word order, and morphologies. LSA decides the meaning of a sentence using the word it contains, and the meaning of a word using the sentences that contain the word. Then, interrelations between words and sentences were discovered with singular value decomposition (SVD). The outline of **latent semantic analysis** is as follows.

1. Extract terms from sentences
2. Build input matrix with an approach on the terms, e.g. tf-idf or binary representation

3. Compute singular value decomposition (SVD) on input matrix
4. Compute ranks based on sigma and V_T from SVD
5. Pick top-N sentences as the summarization

5. GAP SELECTION

The gap selection module utilizes two main NLP components: constituent parsing (using the PCFG Stanford Parser [8]) and an in-house named entity recognition (NER). Constituent tree from syntactic parser is used to determine parts of selected sentence S as the candidate answers $\{a_j\}$. Gap selection accepts a sentence (as a string), and produces fill-in-the-blanks question-answer pairs.

5.1. Named Entity Recognition

The purpose of named entity recognition is to locate and classify named entities from the unstructured text into predefined categories. There are two different approaches in our NER, namely supervised learning and deep learning. The dataset used to train the NER only consists of three useful tags, namely person, location, and organization. To enable time tagging, we integrate another system [8] with our NER, which uses a rule-based datetime finder to extract datetime entity from the text.

Supervised learning approach in NER is done using several features and Conditional Random Field (CRF) classifier. CRF classifier is used because it outperforms other sequence modeling models like Hidden Markov Model (HMM) based on the previous work [9]. The following is the list of features used for supervised learning.

1. Word (normal and lowercase version)
2. Character level n-gram of the word, consists of the first and last 2-3 letters
3. Orthographic features: title, digit, and uppercase
4. POS tag and its character level n-gram (first two letters)
5. Word position from the beginning and end of the sentence
6. Bag-of-word, with window size equals 2 (1 left, 1 right), each window consists of the lowercase word, orthographic features (except digit), POS tags, and character level n-gram of the POS tags (first two letters)
7. Word embedding, using several models like Word2Vec CBOW, Word2Vec SG, and FastText

Deep learning approach in NER is done using bidirectional long short-term memory (Bi-LSTM) CRF architecture. We have two options of models in this approach, the first is built using sequence tagging with Anago [10], and the second is developed using Keras [11] with the detailed implementation as follows.

1. During training the model:
    a. Create a dictionary by converting words in the training data to numeric
2. During testing:
    a. Convert the word processed to numeric based on the dictionary
    b. If word processed by the model is not in the dictionary, convert it to 0
3. The detailed architecture of the Bi-LSTM CRF:
    a. Embedding layer
    b. Bi-LSTM layer
    c. TimeDistributed Dense layer
    d. CRF layer

5.2. Gap Selection

Utilizing the constituent tree and BIO encoding labels from NER, phrases from the sentence are selected as candidate answers. Candidate answers are noun phrases that act as a subject, an object, or a complement in the prepositional phrase of a declarative clause.

Furthermore, there exists a possibility that a candidate noun phrase only consists of a part of a named entity. This is because each word is, by itself, a valid noun phrase. More specifically, if a candidate answer a_j is a part of named entity and a_j does not include all corresponding words needed to form a proper named entity, a_j is eliminated from the set of candidate answers $\{a_j\}$. In the end of the process, the candidate answers are passed to the next module as the answers $\{A_j\}$ of S.

6. QUESTION FORMATION

As with gap selection, question formation is mostly rule-based. Question formation uses the named entities (and their tags) and the constituent tree produced by gap selection, as well as a NLG module [12][13] to convert fill-in-the-blanks question-answer pairs to proper, interrogative-form question-answer pairs. A detailed explanation of each step taken is follows.

6.1. Conversion to Interrogative Form

Depending on the role of the answer in its clause, there are three cases for converting a declarative sentence to its interrogative form.

1. Answer as subject
   Converting a statement to a question when the answer is its subject is simple: replace the subject answer with the question word, and add a question mark to the end of the sentence.
2. Answer as object
   Converting a statement to a question when the answer is its object is considerably more difficult. The constituent tree is traversed top-down with recursive descent and subtrees are used as input for the NLG module, depending on its constituent tag. The answer object is elided from being inputted to the NLG; instead, the NLG generates an interrogative sentence.
3. Answer as prepositional phrase
   Similarly, converting a statement to a question when the answer is its object is difficult. The constituent tree is traversed top-down with recursive descent and subtrees are used as input for the NLG module, depending on its constituent tag. The NLG generates an interrogative sentence with a generic question word; then, the answer is manually removed from the question sentence.

6.2. Determining Question Words

The question words for an interrogative sentence is determined by the NE tag of its answer. For answer as subject or answer as object, the only possible question words are "What" and "Who", as both roles only allow nouns. For answer as prepositional phrase, possible question words are "Who", "Where", "When", and "What", depending on if the answer is, respectively, a person, a location, a time preposition, or anything else.

7. EXPERIMENT

The following section reports the evaluation results of each testable module in the question-answer generation system.

7.1. Dataset

We used two datasets, SQuAD2.0 for the question-answer generation system in general, and CoNLL2003 was specifically used for named entity recognition (NER) [14][15]. Stanford Question Answering Dataset (SQuAD) is a dataset that includes questions generated by crowdworkers and the corresponding passages [14]. However, we only utilized the first paragraph of the passages (without the questions) in the training data as the input to our system. CoNLL2003 (English) is an annotated collection of news wire articles from the Reuters Corpus [15]. The tags used in our system were the names of persons, organizations, and locations, as explained in Section 5.

7.2. Evaluation on Sentence Selection

Summaries generated by system using different methods in sentence selection were evaluated by cosine similarity in Table 1. The evaluation was done with 10 passages from SQuAD dataset, each of them was at least 10 sentences long, with three-sentence summaries.

Table 1. Sentence selection evaluation

Method	Averaged cosine similarity	
	No stopwords	**With stopwords**
LSA (tf-idf)	0.481	0.536
LSA (binary representation)	0.719	0.724
MWPE (W = 5, K = 7)	**0.783**	0.741
TextRank (Jaccard)	0.694	0.698
TextRank (cosine)	0.669	0.593

Based on averaged cosine similarity, it can be seen that almost all of the methods performed better when they considered the presence of stopwords. Stopwords are used to eliminate insignificant words from the processed sentences. The only ones that did not experience increase in evaluation were MWPE and TextRank with cosine similarity. Multi-word phrase extraction with tuned hyperparameter remained on the first position in the evaluation, both with and without stopwords, even though its score decreased by 0.042 because of the usage of stopwords.

7.3. Evaluation on Named Entity Recognition

The best model in **supervised learning** is achieved using tuned c1 and c2 hyperparameters with all the features explained in Section 5. During the feature importance experiment, it can be seen that all features give some contributions to the model, with the most impactful feature is bag-of-words feature. This is because the feature gives context to the model when it is classifying a word. The least impactful feature is position feature, because there are many words in one sentence and there is no pattern for each sentence in the dataset. Word2Vec CBOW, Word2Vec SG,

and FastText are trained using train dataset of CoNLL2003. However, word embedding features obtained do not prove to be useful. This is probably because the word embedding features of each word fail to represent the true meaning of the word, and the word embedding model may face out of vocabulary problem when performing on the test dataset.

In **deep learning** approach, it is shown in Table 2 that Bi-LSTM CRF model developed using Anago outperforms Keras model. This happens because the optimization metric used to train the model in Keras is accuracy. Keras model cannot use f1-score metric, because it has been removed from Keras.

Table 2. Named entity recognition evaluation

Method	f1-score
Supervised learning (tuned hyperparameters)	**0.828**
Deep learning (Keras)	0.313
Deep learning (Anago)	0.760

However, CRF model in supervised learning is able to surpass the deep learning models. It is possible that it is due to the simplicity of features used to develop the models. Feature used in deep learning approach are simply numbers converted from words, so it is possible that the feature cannot represent the true meaning of data and fail to become discriminative features. Hence, the supervised model is used as the final NER model due to its high performance.

5. RESULT

Figure 2 shows sample question-answer pairs generated by our proposed system, as well as its context sentence. The first five question-answer pairs show successful question generation; the final pair, however, presents a failed question generation.

Questions generated using our system successfully converted simple declarative statements into questions, but for more complex sentences, our devised rules show difficulties in converting more complex statements to questions. This is due to an implementation error—the rules to convert a statement to a question are made in-house, instead of implementing an already established rule-based question-generation system, e.g., Heilman and Smith [16].

Another major limitation of our system is its limited pool of possible question words, i.e., only WH-questions are generated, and only questions which do not require complex semantic analysis (e.g. “Why”, “How”) can be generated.

Our question-answer generation was implemented both in the form of Jupyter notebook and simple web application. The web application version in Figure 3-6 provides ease of use and user-friendliness, while the Jupyter notebook version focuses more on customization and flexibility needed to reproduce the system.

Sentence: The building was on fire, and it wasn’t my fault.
Question: What was The building on?
Answer: fire

Sentence: Distant streetlights provided the only light in the dusty hall, and left huge swaths of blackness crouching in the old classroom doors.
Question: What provided the only light in the dusty hall, and left huge swaths of blackness crouching in the old classroom doors?
Answer: Distant streetlights

Sentence: I sprinted around a corner and toward the exit doors to the abandoned school building on the southwest edge of Chicagoland.
Question: Where did I sprint around a corner and toward the exit doors to the abandoned school building on the southwest edge of?
Answer: Chicagoland

Sentence: The movie of my life must be really low-budget.
Question: What must be really low-budget?
Answer: The movie of my life

Sentence: There is, I think, humor here which does not translate well from English into sanity.
Question: What does does not do?
Answer: sanity

Fig 2. Sample generated question-answer pairs

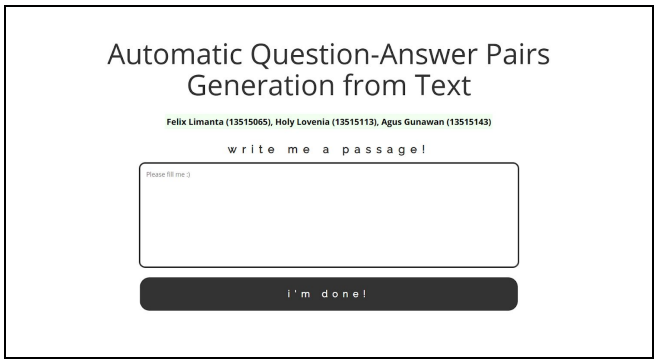

Fig 3. QAG web application in initial state

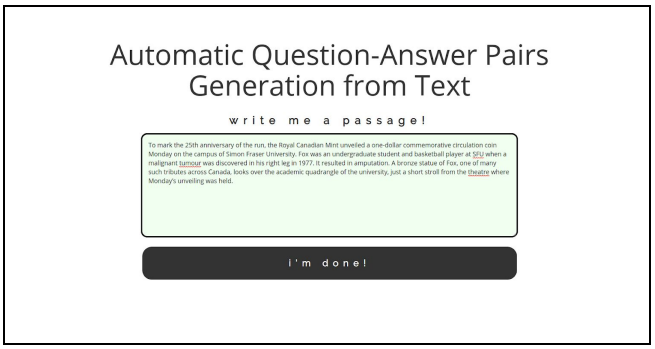

Fig 4. QAG web application with passage input

Fig 5. QAG web application with QA results

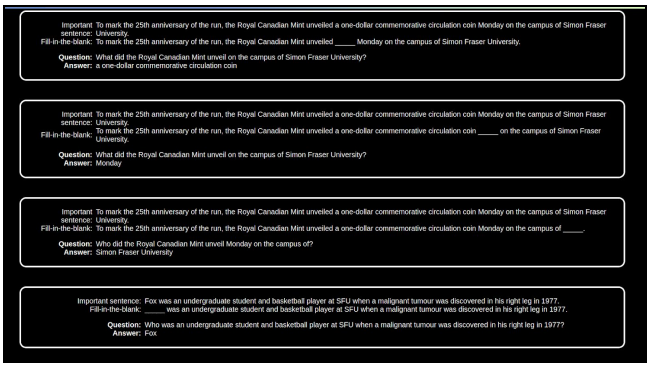

Fig 6. QAG web application with more QA results

6. CONCLUSION AND FUTURE WORK

We have presented a rule-based automatic question generation system for reading comprehension. We use multiple methods to select noteworthy sentences in a paragraph, then use named entity recognition and constituent parsing to generate possible question-answer pairs. The sentence is then transformed to an interrogative form based on a set of rules and possible answers. The system performs well on simple sentences, but falters on more complex sentences. Our question-answer generation system is able to produce three types of WH-questions, e.g. "What", "Where", and "Who".

Future improvements to the system include implementing a more robust, already established rule-based question-generation system, e.g., Heilman and Smith [16]. Another option is to implement a deep neural network architecture for the question generation itself, e.g., using sequence-to-sequence.

7. REFERENCES

[1] E. Reiter and R. Dale, *Building Natural Language Generation Systems*. Cambridge University Press, 2000.

[2] D. L. Lindberg, *Automatic Question Generation from Text for Self-directed Learning*. 2013.

[3] X. Du, J. Shao, and C. Cardie, "Learning to Ask: Neural Question Generation for Reading Comprehension," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

[4] A. Sarvaiya, "Using Natural Language Processing for Smart Question Generation," *Intel AI Academy*, Jul. 2018.

[5] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[6] L. Page and S. Brin, "The PageRank Citation Ranking: Bringing Order to the Web," 1998.

[7] J. Steinberger and K. Ježek, "Using Latent Semantic Analysis in Text Summarization and Summary Evaluation," *Proceedings of the 7th International Conference ISIM*, Jan. 2004.

[8] akoumjian, "akoumjian/datefinder," *GitHub*. [Online]. Available: https://github.com/akoumjian/datefinder. [Accessed: 10-Nov-2018].

[9] M. Agarwal *et al.*, "Comparative Analysis of the Performance of CRF, HMM and MaxEnt for Part-of-Speech Tagging, Chunking and Named Entity Recognition for a Morphologically rich language," Nov. 2018.

[10] Hironsan, "Anago," *GitHub*. [Online]. Available: https://github.com/Hironsan/anago.

[11] F. Chollet, "Keras," *GitHub*, 2015. [Online]. Available: https://github.com/fchollet/keras.

[12] S. Mahamood, "SimpleNLG," *GitHub*. [Online]. Available: https://github.com/simplenlg/simplenlg.

[13] B. Dagenais, "py4j," *py4j*. [Online]. Available: https://www.py4j.org.

[14] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000 Questions for Machine Comprehension of Text," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

[15] E. F. Tjong Kim Sang, E. Tjong Kim, and F. De Meulder, "Introduction to the CoNLL-2003 shared task," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003* -, 2003.

[16] M. Heilman and N. A. Smith, "Question Generation via

Overgenerating Transformations and Ranking,” 2009.