Question Generation using NLP

Kaksha Mhatre kakshapmhatre@gmail.com
Akshada Thube akd.thube@gmail.com
Hemraj Mahadeshwar mahadeshwar.hemraj97@gmail.com
Prof. Avinash Shrivas
Dept. of Computer Science
Vidyalankar Institute of Technologyy of University
Mumbai, MH, India

***Abstract* - In this paper, we have proposed a method to question generation using coreNLP and tree regular expressions for parse tree manipulation. In addition to this we have used spaCy for named entity recognition to recognize entities for question word identification.**

***Keywords*-Question generation, coreNLP, spaCy.**

I. INTRODUCTION

Question generation is a way by which we can generate questions from a given text file automatically. It is a process in which the input is a text file containing a paragraph text and the output is a set of questions which can be generated on the paragraph. Question generation has various applications like MCQ generation, GRE paper setting, frequently asked questions (FAQs), intelligent tutoring systems, English Comprehension question generation, etc.

Automating the question generation (QG) process can help reduce the workload and dependency on humans; thereby reducing the human error. The automatic question generation is an important research area which is potentially useful in dialogue systems, instructional games, educational technologies etc.

II.RELATED WORK

In some earlier work on question generation, Heilman and Smith used general-purpose rules to transform sentences into questions and Sneider used templates. In this paper, we present a system which uses syntactic and keyword modeling. By contrast, we use phrase-specific question generation. The system will take a paragraph as input and generate important questions from the sentences extracted from the paragraph. It will generate different types of WH-questions from those selected sentences as well as Yes-No type questions. The questions will be generated from both simple and complex sentences. We use coreNLP for Tregex and spaCy for efficient question word identification as it has more entity types.

III. PROBLEM DESCRIPTION

In this paper, we used an approach based on syntactic and keyword modeling for generating questions. In particular, we used parse tree manipulation as well as named entity recognition to generate factoid questions from input documents. We have described how to generate different types of question from a single input sentence, including Yes-No, Who, What, Where and How questions. We have presented the evaluation for our factoid question generation method, and we also have discussed our plans to use question generation for question answering. A question generator for a QA system will determine the highly probable questions that a given document can answer.

IV.PROPOSED METHODOLOGY

First, the input text that is the paragraph is split into single sentences, then using appropriate question word, we generate questions. From a single sentence, we produce yes-no questions over the sentence and WH-questions from the subject, object, adverbials, and prepositional phrases in the sentence. In this paper we describe a QG system which can generate appositive questions and subject-object questions from an input text document.

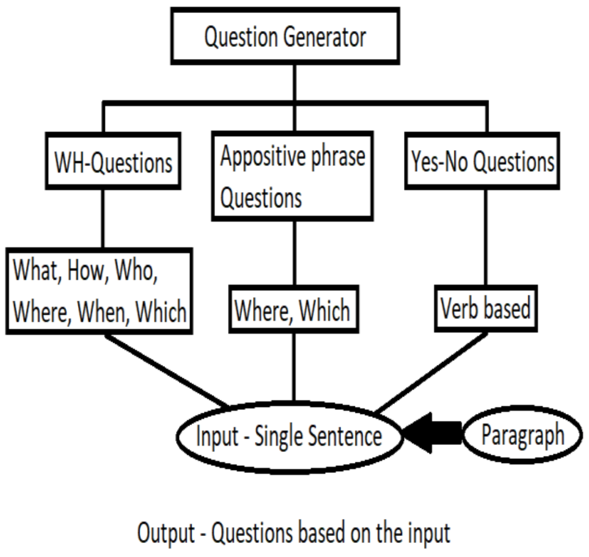

Fig.1 Proposed methodology for question generation

This paper is organized as follows:
Section V deals with question generation which includes the pre-processing stage and question generator module. Section VI presents a conclusion of our method and our

ideas for future work, including using question generation for question answering.

V.IMPLEMENTATION

In our factoid question generation system, individual sentences are used from an input text document. Firstly, we pre-process each sentence, parsing it to perform anaphor resolution as well as pronoun replacement. Hence, complex sentences are divided into independent clauses. This is described in Section A.

After this, a question generator module is used so that each independent clause passes through it and outputs one of more questions from that clause. The question generation module has access to sentence simplification processes as well as to a question word identification process. These processes are described in Section B. The actual process of generating output questions is described in Section C. To manipulate parse trees, we use coreNLP for Tregex, a tree query language, and Tsurgeon, a tree manipulation language. In Section D, we have displayed the output of the question generator.

1. Sentence pre-processing

Questions generated from large complex sentences can sometimes be meaningless which is why we need to pre-process them. In the pre-processing stage, complex sentences are divided into simpler sentences.

1.1 Parsing

Firstly, we parse each input sentence using the Stanford parser. This parser outputs constituency structures which we manipulate in later processing.

1.2 Extracting Independent Clauses from Complex Sentences

We split one type of complex sentence: those formed by joining simple independent clauses using conjunctions. For example, the complex sentence.

1.3 Shriya went to the mall and she bought some clothes and she tried them

This sentence consists of three independent clauses joined by the conjunction and after sentence splitting, we obtain three separate sentences

1.4 Shriya went to the mall. She bought some clothes. She tried them

The Tregex expression we use to identify independent clauses in an input parse tree is:S=n1 $ (CC > (S >> ROOT)) !< (S $ (CC > (S >> ROOT))) The separated independent clauses are sent to the question generator module.

1.4 Resolving Anaphors and Replacing Pronouns-A complex sentence includes pronouns and simply splitting the sentences gives ambiguous or unclear output. Anaphora resolution and pronoun replacement enhance the quality of questions generated from such sentences. The Hobbs algorithm is implemented here to perform pronoun resolution. Anaphora resolution and pronoun replacement are performed on the input text as a part of pre-processing. The output from sentence splitting including pronoun replacement for our example sentence is

1.5 Shriya went to the mall. Shriya bought some clothes Shriya tried them- As a result more meaningful questions like 'What did Shriya buy?' Instead of 'What did she buy?' can be formed.

2. Additional Sentence pre-processing

In some of the question generation modules, we use the following processes question word identification; subject-auxiliary inversion; appositive removal and prepositional phrase removal.

2.1 Question Word Identification

The Stanford Named Entity Recognizer (NER) is used in this to find the entity type of each answer phrase we find. The Stanford named entity recognizer outputs three entity types as well as NO ENTITY: PERSON, LOCATION, and ORGANIZATION. In addition to this, we used spaCy which is able to identify many more entities which help us identify question word more accurately. We select the question word using the identified entity type. We use the following algorithm to select question words:

- If there is a proper noun phrase (phrase type nnp) in the parse tree of the answer phrase then we pass the proper noun phrase into the nor. if the nor outputs person or organization we use 'who/whom'; if it outputs location we use 'where'. for location 'which' can also be an appropriate question word.
- If there are no proper nouns present then we remove stop words from the answer phrase and pass the remaining words into nor. if the nor outputs person or organization we use 'who/whom'; if it outputs location we use 'where'. if the output from ner is no entity then we use the question word 'what'.If the answer phrase is an adverb then we use the question word 'how'. Often, wrong questions result if the ner makes a mistake in identifying the correct entity type or if it fails to identify an entity type for a phrase. An example is the question Where was Abdul Kalam? Generated from the sentence Abdul Kalam was the president of India. This happened because when the Object (NP) President of India was passed to the NER it returned LOCATION as one of the entity types because of the presence of the word India (the NER also output NO ENTITY). Hence the question word chosen was 'WhereSubject-Auxiliary Inversion (SAI)

2.2 Subject-Auxiliary Inversion (SAI)

*S*ubject-auxiliary inversion is used to construct Yes-No questions, and to construct the input to the Object NP question generation process, the adverbial question generation process, and the prepositional phrase question generation process. Subject–auxiliary inversion is the placement of an auxiliary verb in front of the subject. If an auxiliary verb is not present an appropriate form of 'Do', like 'Do', 'Does' or 'Did' is inserted. We do not perform 'Do insertion' if the main verb in the sentence is a form of the verb to be. When we do perform 'Do insertion' we also reduce the main verb of the sentence to its base form using coreNLP (Tregex, Tsurgeon) and Word Net.

2.3 Appositive Removal

An appositive is a noun or a noun phrase adjacent to another noun phrase, which explains or defines the noun it follows and is typically set off from it by a comma. For example, the sentence below contains one appositive, the biggest city in the world Mexico City, the biggest city in the world, has many interesting archaeological sites. The removal of an appositive from a sentence does not alter the meaning of the sentence in any way. For example, if we remove the appositive from the sentence above we get Mexico City has many interesting archaeological sites.We identify appositives from input sentences using the following Tregex expression: SBAR|VP|NP=app $, /,/ We can also generate questions from appositives although we usually use appositive removal as a form of sentence simplification.

2.4 Prepositional phrase removal

Prepositional phrases in sentences are usually not crucial to the meaning of the sentence, and may unnecessarily increase the length of generated questions. So we use prepositional phrase removal as another form of sentence simplification.We identify prepositional phrases using the following Tregex expression- PP = n1 >> (S > ROOT) < NP !.. PP !<< PP .

3. Question Generation

After each sentence is pre-processed, question generation is performed on each independent clause. Multiple questions can be produced from each independent clause: we generate Yes-No questions, as well as WH-questions from the subject and object of the clause and from adverbials, appositives, and prepositional phrases in the clause. The Yes-No question generation, Subject NP question generation, and appositive question generation modules take input clauses directly from the pre-processing module, while the other modules take input from after subject auxiliary inversion.

3.1 Yes-No Question Generation

A Yes-No question is a question whose answer is a Yes or a No. Such a question does not contain any question word (What, Where, Who, How etc). Yes-No questions are created by performing subject auxiliary inversion. For example, if the input clause is She ate the fruits. Then the Yes-No question generated is Did she eat the fruits?

3.2 Generating Questions on Subject NPs

English language sentences follow a Subject Verb Object (SVO) structure. So to find the subject of a simple declarative clause we simply find the NP that starts the sentence and precedes the verb. To generate a question on the subject we extract the subject NP, find its corresponding entity type, and join the question word to the sentence's main verb. The Tregex expression we use to identify the subject NP is NP = n1 > (S = n2 > ROOT) & $++ VP = n3.For example, if the input clause is Barack Obama is the president of America, then the question generated on the subject is: Who is the president of America?

3.3 Generating Questions on Object NPs

Transitive English sentences contain objects on which the verb acts. In a simple declarative clause, the object follows the verb phrase. We generate questions on the objects of clauses that have undergone subject auxiliary inversion. To generate a question on the object of a sentence we extract the object NP, find its corresponding entity type, and join the question word to the front of the sentence. The Tregex expression we use to identify the object NP is:NP=n1 !>> NP >> (VP > (S=n2 > ROOT)) For example, if the input clause is Dianna liked Jake. Then the question generated on the object is: Who did Dianna like?

3.4 Generating Questions on Appositives

- To perform question generation on appositives, we use the following algorithm:
- Extract appositive from the input sentence.
- Perform question word identification on the NP, VP, or SBAR preceding the appositive.
- Join the question word with the appositive to generate an output question.

  For example, if the input sentence is: Mexico City, the biggest city in the world, has many interesting archaeological sites. Then the question generated on the appositive is:Which/Where is the biggest city in the world? {Answer:-Mexico City}

3.5 Generating Questions on Prepositional Phrases

- To generate questions on prepositional phrases we need the output of the Subject Auxiliary Inversion module. The algorithm for generating questions is then.
- Extract each prepositional phrase and find the object of the preposition.
- Find the question word for the object of the preposition.
- Replace the object of the preposition with the question word.

3.6 Relocate the prepositional phrase to the start of the sentence- The Tregex expression used to identify PP is given below PP = n1 >> (S > ROOT) < NP !.. PP !<< PP If the input sentence is The dog is asleep on his bed. Then the question generated from the only prepositional phrase (on his bed) is On what is the dog asleep? Generating Questions on Adverbials

We generate questions on adverbials from the output of the Subject Auxiliary Inversion module. The algorithm for generating questions is then:

- Find each adverbial and remove it from the sentence.
- Add the question word How to the start of the sentence.

  The adverbial is found using the Tregex pattern RB=n1 > (ADVP >> (S=n2 > ROOT)) | > (ADJP >> (S=n2 > ROOT)) If the input sentence is:

The meeting went well.Then the question generated from the adverb well is How did the meeting go?

4. Running the Question Generator

We conclude this section with two examples of input sentences and the questions our factoid question generator can produce from them. Given the input sentence Mexico City, the biggest city in the world, has many interesting archaeological sites . Our factoid question generator outputs Yes-No: Does Mexico City, the biggest city in the world, have many archaeological sites? {Answer:-Yes} Subject: Which/Where has many archaeological sites? {Answer:-Mexico City, the biggest city in the world}Object: What Does Mexico City, the biggest city in the world, have? {Answer:-many archaeological sites} Appositive: Which/Where has many archaeological sites? {Answer:-Mexico City} Prepositional phrase: In what does Mexico City, the biggest city, have many archaeological sites? {Answer:-the world}.

VI.CONCLUSION AND FUTURE WORK

Our question generation system always produces more questions than the number of questions generated by the human judges. There is scope of improvement of the pre-processing stage, which is responsible for converting long complex sentences into small simpler sentences. For example, we should add functionality to simplify sentences like John went to the market, bought some fruits and ate them. To generate questions containing the question word Why, How much, To what extent etc, we used an improved named entity recognizer spaCy which is capable of labeling more entity types, including money, dates/times, etc. Finally, we plan on integrating this question generator with a QA system. If an input question finds a match with a question produced by the question generator, then the QA system can return the corresponding phrase or sentence rather than searching the document database. In this paper we presented an approach to question generation using parse tree manipulation, named entity recognition using coreNLP as well as spaCy. We described an efficient question generation method for generating factoid questions. We displayed how our question generation approach was able to generate multiple questions from a single input sentence.

REFERENCES

[1] Saidalavi Kalady, Ajeesh Elikkottil, Rajarshi Das, "Natural Language Question Generation Using Syntax and Keywords", 2010.
[2] Michael Heilman,Noah A.Smith," Extracting Simplified Statements for Factoid Question Generation", 2011.
[3] Sneiders, E. (2002). Automated question answering using question templates that cover the conceptual model of the database. In Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems (pp. 235-239). K. Elissa, "Title of paper if known," unpublished.
[4] Hobbs, J. (1977) 38 examples of elusive antecedents from published texts (Research Report #2), New York: Department of Computer Sciences City College, City University of NewYork Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[5] Heilman, M., & Smith, N. (2009). Question generation via overgenerating transformations and ranking. Technical Report CMU-LTI-09-013, Carnegie Mellon University