Potential Question Generation – Project Report

Isabelle Nguyen

Introduction

This report serves to illustrate my project implementation of Question Generation which is informed by Edgar Onea‘s thesis „Potential Questions in Discourse and Grammar“. The project was implemented in Python and uses the Spacy and NLTK libraries. The structure of this report is as follows: First, I will recap some theoretical background from Semantics/Pragmatics. I will then present the idea from Onea‘s thesis essential for this project. Onea‘s principles for question generation will be presented along with a description of my implementations. I will then look at the questions generated by the system and point out good examples, as well as failures. The last part will be an error analysis and an outlook on what could be improved in the next version.

Theoretical Background

The idea of questions as discourse-structuring devices was first formulated by the proponents of the QUD („Questions under Discussion“) framework. (cf. Roberts 1996) According to this theory, the purpose of all discourse is to answer a constantly changing set of QUDs which are hierarchically organized, from the general to the specific. Questions are therefore more than just one element of discourse because without them there would be no discourse at all. This is an interesting topic from the perspectives of various fields, such as Semantics, Pragmatics, Philosophy, Computational Linguistics and Artificial Intelligence. If questions are such an essential element of human discourse, then a machine whose purpose is to imitate human behaviour should be good at asking – and not only answering – questions. However, since the QUD framework concerns itself with making explicit the question currently under discussion, its practical applications are rather limited and pertain mainly to comprehension assessment. (cf. Heilman 2011) This is where Onea‘s framework of Potential Questions comes into play. (cf. Onea 2013) Just like the proponents of QUD, Onea thinks that discourse (whether oral or written) is organized by questions. However, rather than looking at the question currently under discussion (i.e. the implicit question that generated the current utterance), Onea is interested in the questions that are opened up by the current utterance. This is a promising approach for dialogue systems and other machine applications designed to mimic human discourse strategies.

Potential Question Generation

Onea‘s framework of Potential Questions exploits a notion from Inquisitive Semantics, where utterances can be described in terms of their informative and their inquisitive value. In this framework, propositions that display both characteristics can become questions (cf. Onea p. 327). Formally,

> φ is a potential question licensed by an assertion ψ relative to a discourse state σ, iff φ is a question in $\psi \wedge !\sigma$ and φ is not a question in the discourse state σ. (p. 326)

The licensing of a potential question thus consists of two components: Its congruency with respect to the current assertion as well as to the discourse as a whole. While the solution of the latter – which would require some sort of knowledge database – is not feasible in the scope of the project presented here, the former – Question Generation with regard to a single utterance – has been the focus of this project.

It is important to note that Onea did not design his framework for an „automated or half-automated procedure“ (p. 326) Therefore, it can not be expected that all of his generation principles and constraints will be implemented. In the following subsection I will present Onea‘s principles as well as information about how I implemented them in my Question Generation system.

Question Generation Implementation

1. Follow formal hints

According to this principle, one should look out for elements that are underdetermined, such as indefinite determiners. This is easily implemented by looking for such determiners in the sentence and ask for a specification. In my system, one method is responsible for subjects, the other one for objects. Both look for underspecified constituents and generate *Which*-questions.

2. Unarticulated constituents

As with other principles, this one is formulated rather vaguely, but from the examples that Onea provides, it can be deduced that verbal arguments that are not expressed in the sentence should be asked about. This is not a trivial task as it requires information about a verb‘s subcategorization frame. NLTK‘s Propbank binding proved to be a valuable resource. The system‘s method looks at

all the verbs in the input sentence and counts the objects. If the verbs‘ subcategorization frames require an object argument but none is present, a *What*-question is generated. Similarly, if a *MNR*[1]-object is part of the frame, a *How*-question if asked. Finally, if the verb takes prepositional objects, the method checks if the relevant prepositions are present in the sentence, and if they are not, the corresponding questions are generated, with both *What* and *Who* as question words.

3. Indexicals

Again, it is not entirely clear what this principle asks for. In his explanation, Onea writes that „whenever the indexicals are not clarified by the context“, one should „ask questions of clarification about them“ (p. 332). However, one of the two examples he gives are not about an indexical from the sentence at all. My system therefore generates a *Where*-question whenever there is an indexical *there* in the sentence or if the verb‘s subcategorization frame includes a prepositional object with *at* and that preposition is not realized in the sentence. Also, a *When*-question is generated whenever there is no information about time in the sentence. This implementation exploits the Named Entity information provided by the Spacy annotation tool.

4. Rhetorical relations

Rhetorical relations are discourse-structuring relations between sentences such as Elaboration, Result, Justification. Of these, Justification is arguably the most prominent, as *Why*-questions are extremely common in discourse. The system implements *Why*-questions as well as questions about the source of an information. The implementation of other rhetorical relations did not seem feasible to me.

5. Parallelism and Contrast

This principle can be described as finding rhetorical patterns in discourse and asking for their completion. Since this is a principle that works at the discourse rather than at the sentence level, it could not be implemented in the present project.

6. Animacy Hierarchy

Ask for more information about animated individuals (e.g. persons). This was implemented for entities that are recognized as PERSON or ORG (organisation) by the Named Entity parser. Unfortunately, however, in the current implementation, it could not be applied to noun phrases that are not Named Entities but still refer to persons, like *the women*.

1 For *manner* – how is something done?

7. Mystery

This principle tells us to ask about unusual events or relations. I implemented it by asking about any underspecified object. It is therefore included in the object method described above.

In order to preserve grammaticality, the generation of all *wh*-questions is preceded by the following actions:

- Change of second and first person pronouns and auxiliaries
- Auxiliary generation
- Inversion of subject and auxiliary

In addition, the information garnered by parsing the input sentences for Named Entities is exploited by copying those entities into corresponding question templates that were scraped from the Yahoo question corpus. This is justified by Onea‘s observation that „one can imagine texts in which [seemingly absurd questions] become perfectly natural“ (p. 328), and they can provide surprisingly fitting questions in some contexts.

Also, since input questions can be much more complex than those encountered in the training set, I have implemented various functions whose purpose is to break down long sentences into shorter sequences. We will see that this does work sometimes, but is not always enough to capture the complexity of natural language.

Question Generation in Practice

In this section, I will look at the questions generated by the system. In his thesis, Onea provides examples of input sentences along with the target questions. I will check whether my system generates those questions. In addition, I will compare the automatic question generation to human-annotated questions for an interview with Edward Snowden[2]. The questions were generated as QUDs but it can be argued that in a coherent discourse, the Question under Discussion that governs the present utterance is a potential question generated by the previous utterance. The interview extract is used as a way of validating how well the system can handle more complex sentences. All

2 Sentences were slightly modified to produce well-formed input, e.g first words were capitalized, contracted forms were spelled out etc.

input sentences and their corresponding questions can be found in the accompanying Jupyter Notebook.

For Onea‘s test sentences, 7 out of 14 were generated correctly. The reasons for not generating the target questions are the following:

a) A non-personal noun is not recognized as an entity with the feature [+human], thus the desired question „What else do we know about the man?“ is not generated upon inputting „A man is eating.“ This is because the animacy method relies on information from the Named Entity parser, as described above. For the same reason, the sentence „John saw a chair“ does generate the target question „What else do we know about John?“ - the Named Entity tag PERSON that is attached to *John* triggers the generation of the question.

b) Some target questions require external knowledge that the system does not have. For example, the sentence „Peter has eaten“ is supposed to generate „Did he have enough?“ and „Did he use a spoon?“. Similarly, „Peter is eating a sandwich“ should generate „Is the sandwich with ham?“. My system can not generate any of those questions.

c) The third type of mistake pertains to flaws in the tagging systems. For instance, the verb *eat* is recorded in the Propbank as having only an agentive argument and a local argument, which is why, for the sentence „Peter is eating“, it generates „Where is Peter eating?“, but not the target question „What is he eating?“. In contrast, the sentence „My sister was reading“ does trigger the question „What is your sister reading?“, because the Propbank‘s entry for *read* includes a direct object.

While in the last example, a target question is simply not generated, faulty tagging can result in the generation of nonsensical and ungrammatical questions as well, as can be observed in one sentence of the Snowden example. The sentence „Because, I was reading that you asked for a kind of police protection“ is supposed to generate the question „Why did you ask for police protection?“. But the Spacy POS module wrongly tags the subordinating conjunction *that* as the direct object of the sentence. When changing the sentence minimally to „Because, I read that you asked for a kind of police protection“, it is, however, tagged correctly, and the question „Why did you ask for the kind of police protection?“ is generated. All in all, the Snowden sentences prove to be much harder for the system to dissect than the shorter and more simple test sentences. As a result, many of the questions are ungrammatical. Most prominently, the system can not deal with expletive subjects. Therefore, the sentence „There are significant threats but I sleep very well“ triggers the question „Which there are significant threats?“. Also, it is unable to resolve coreference between relative pronouns and their antecedents, and has problems with relative subjects and objects in general.

Conclusion

The current version of my Question Generation system can generate questions that do not require any world knowledge, and also some that do, thanks to the use of question templates and Named Entity Recognition – when presented with rather simple utterances. However, when the input consists of more complex sentences, it is prone to form grammatically wrong and/or nonsensical questions. Some of these flaws should be easily repairable with some additional work. The next steps would be:

- Exploit the information given by the Propbank even more and fine-tune questions about arguments
- Implement a relative subject/object and coreference resolution
- Deal with expletive subjects
- Search for more robust POS and Named Entity taggers
- Implement stronger modularization for more flexibility.

References

BIRD, STEVEN, EDWARD LOPER and EWAN KLEIN. 2009. Natural Language Processing with Python. O‘Reilly Media Inc.

BONIAL, CLAIRE, OLGA BABKO-MALAYA, JINHO D. CHOI, JENA HWANG and MARTHA PALMER. 2010. PropBank Annotation Guidelines

HEILMAN, MICHAEL. 2009. Question Generation via Overgenerating Transformations and Ranking

HEILMAN, MICHAEL. 2011. Automatic Factual Question Generation from Text

HONNIBAL, MATTHEW and INES MONTANI. 2017. Spacy2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing

ONEA, EDGAR. 2013. Potential Questions in Discourse and Grammar.

ROBERTS, CRAIGE. 1996. Information Structure in Discourse. Towards and Integrated Formal Theory of Pragmatics