Proceedings of QG2010: The Third Workshop on Question Generation

Edited Book

QG2010: The Third Workshop on Question Generation

Including:
QGSTEC2010: The First Question Generation Shared Task and Evaluation Challenge

At The Tenth International Conference on
Intelligent Tutoring Systems (ITS2010),

Carnegie Mellon University, Pittsburgh, Pennsylvania, USA,

June 18, 2010

Editors: Kristy Elizabeth Boyer and Paul Piwek

Introduction

Asking questions is an important function of advanced learning technologies such as intelligent tutoring systems, inquiry-based environments, and game-based learning environments. Automatically generating high-quality questions is the long-term goal of the emerging Question Generation (QG) research community, which brings together researchers from a wide variety of disciplines including, but not limited to, Intelligent Tutoring Systems, Psycholinguistics, Discourse and Dialogue, Natural Language Generation, and Natural Language Understanding.

QG 2010 is the third in a line of workshops on Question Generation. The Second Workshop on Question Generation was held in conjunction with the 2009 International Conference on Artificial Intelligence in Education, and the First Workshop on the Question Generation Shared Task and Evaluation Challenge was held in 2008 at the National Science Foundation.

The main track of QG2010 hosts five papers, with no less than seventeen different authors from India, UK, and the United States. The papers approach QG from a variety of angles including Question Answering (Kalady et al.), Dialogue Generation (Piwek and Stoyanchev), and Intelligent Tutoring (Sullins et al.). Additionally, some papers focus on addressing specific subtasks for QG, such as Extraction of Simplified Statements (Heilman and Smith) and Target Concept Identification (Becker et al.).

This year, for the first time, the Question Generation workshop hosts a special track with papers on the First Question Generation Shared Task and Evaluation Challenge, QGSTEC 2010. The challenge consisted of two tasks. The first task focused on evaluating the generation of questions from paragraphs and the second on generation from sentences. Five teams entered the challenge from participating institutions in Germany, India, Canada, USA and UK. Each team encouraged the advancement of QG techniques by submitting a paper describing the approach used by their QG system. These approaches will be presented at the workshop.

Acknowledgements

This year's workshop would not have been possible without the enthusiastic help of the QG2010 Steering Committee. Special thanks are due to Vasile Rus for his encouragement and advice. The Programme Committee members and English language mentors have helped ensure the quality of the contributions with their extensive feedback to the authors.

TABLE OF CONTENTS

Main Track

QGSTEC Track

Natural Language Question Generation Using Syntax and Keywords

Saidalavi Kalady[1], Ajeesh Elikkottil[1], Rajarshi Das[1],

[1]*Department of Computer Science and Engineering, National Institute of Technology-Calicut, 673601, Kerala, India*
{said, ajeeshelikkottil, rajarshi.nitc}@nitc.ac.in

Abstract. In this paper we presented an approach to question generation based on syntactic and keyword modeling. In particular, we use parse tree manipulation, named entity recognition, and Up-Keys (significant phrases in a document) to generate factoid and definitional questions from input documents. We describe how to generate different types of question from a single input sentence, including Yes-No, Who, What, Where and How questions. We present evaluation for our factoid question generation method, and we discuss our plans to use question generation for question

Keywords: Question generation, Factoid and definitional questions, Up-Keys.

1 Introduction

The automatic generation of questions is an important research area potentially useful in intelligent tutoring systems, dialogue systems, educational technologies [10], instructional games etc. For example, Gates used question generation to form question-answering exercises for reading tutoring [9]; Harabagiu et al. used question generation in a Question Answering (QA) system to predetermine the questions that can be asked of it [13]. In our previous work we developed an automatic summarizer to aid a QA system [1]. Here we build a question generator for a QA system that will determine the highly probable questions that a given document can answer.

In earlier work on question generation, Sneider [11] used templates, and Hielman and Smith [8] used general-purpose rules to transform sentences into questions. By contrast, we use phrase-specific question generation. 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 both factoid and definitional questions from an input text document. We have developed two different strategies to generate factoid questions and definitional questions. Factoid questions have short fact based answers and are usually generated from a single sentence of the input document. Several factoid questions can be generated from the same input sentence. Definitional questions have a descriptive answer that may be distributed throughout the document. For generating definitional

questions we need to understand the meaning of the whole document, hence we use Up-Keys [1] to generate definitional questions. Up-Keys are keywords of a document and are generated after processing the document using various heuristics given in [1].

The rest of this paper is organized as follows: Section 2 deals with question generation which includes the preprocessing stage and question generator module. Section 3 talks about the definitional question generation method. Section 4 describes the evaluation we performed on the output of the question generator. Section 5 presents a discussion of our method and our ideas for future work, including using question generation for question answering. We conclude in Section 6.

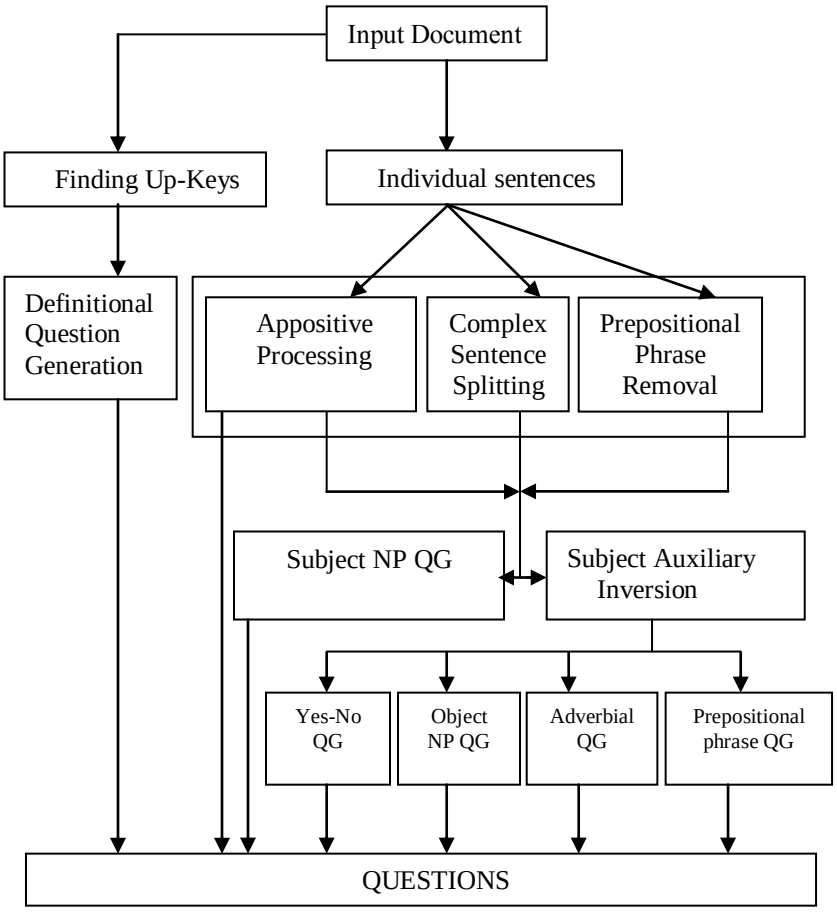


Fig 1. Figure showing the design of the Question Generation system. Individual sentences are processed to generate factoid questions whereas *Up-Keys (keywords)* from the document are used to generate definitional questions.

2 Factoid Question Generation

Our factoid question generation system operates over individual sentences from an input document. First, each input sentence is preprocessed, we parse it, perform anaphor resolution and pronoun replacement, and separate complex sentences into independent clauses. This stage is described in Section 2.1. Second, each independent clause is passed into the question generation module, which outputs one or 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 are described in Section 2.2. The actual process of generating output questions is described in Section 2.3. To manipulate parse trees, we use Tregex [3], a tree query language, and Tsurgeon [3], a tree manipulation language.

2.1 Sentence Preprocessing

In the preprocessing stage we attempt to convert complex sentences into simpler sentence(s). This is done because questions generated from large complex sentences are sometimes meaningless.

2.1.1 Parsing

First, we parse each input sentence using the Stanford parser [2]. This parser outputs constituency structures which we manipulate in later processing.

2.1.2 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

John went to the market and he bought some fruits and he ate them.

consists of three independent clauses joined by the conjunction *and*. After sentence splitting, we obtain three separate sentences:

John went to the market.
He bought some fruits.
He ate 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))) (1)

The separated independent clauses are sent to the question generator module.

2.1.3 Resolving Anaphors and Replacing Pronouns

Sometimes, 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. We implemented the Hobbs algorithm to perform pronoun resolution [12]. Anaphora resolution and pronoun replacement [1] are performed on the input text as a part

of preprocessing. The output from sentence splitting including pronoun replacement for our example sentence is:

John went to the market.
John bought some fruits.
John ate some fruits.

As a result more meaningful questions like *What did John buy?* instead of *What did he buy?* can be formed.

2.2 Additional Sentence 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.

2.2.1 Question Word Identification

We use the Stanford Named Entity Recognizer (NER) [4] to find the entity type of each answer phrase we find. This named entity recognizer outputs three entity types as well as NO ENTITY: PERSON, LOCATION, and ORGANIZATION. We select the question word using the identified entity type. We use the following algorithm to select question words:

1) 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 NER. If the NER 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.
2) If there are no proper nouns present then we remove stop words from the answer phrase and pass the remaining words into NER. If the NER 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'*.
3) 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 Elizabeth?* generated from the sentence *Elizabeth was the queen regnant of England.* This happened because when the Object (NP) *Queen regnant of England* was passed to the NER it returned LOCATION as one of the entity types because of the presence of the word *England* (the NER also output NO ENTITY). Hence the question word chosen was '*Where'*.

2.2.2 Subject-Auxiliary Inversion (SAI)

We use subject-auxiliary inversion 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 Tregex, Tsurgeon and WordNet [6].

2.2.3 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 [5].

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 $, /,/ (2)

We use appositive removal as a form of sentence simplification, but we also generate questions from appositives.

2.2.4 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)

2.3 Question Generation

After each sentence is preprocessed, we perform question generation 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 preprocessing module, while the other modules take input from after subject auxiliary inversion (see Figure 1).

2.3.1 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?

2.3.2 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 (4)

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?

2.3.3 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)) (5)

For example, if the input clause is:

John loved Anne.

Then the question generated on the object is:

Who did John love?

2.3.4 Generating Questions on Appositives

To perform question generation on appositives, we use the following algorithm:

1) Extract appositive from the input sentence.
2) Perform question word identification on the NP, VP, or SBAR preceding the appositive.
3) 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 [5].

Then the question generated on the appositive is:

Which/Where is the biggest city in the world? {Answer:-Mexico City}

2.3.5 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:

1) Extract each prepositional phrase and find the object of the preposition.
2) Find the question word for the object of the preposition.
3) Replace the object of the preposition with the question word.
4) 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 (6)

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?

2.3.6 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:

1) Find each adverbial and remove it from the sentence.
2) 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)) (7)

If the input sentence is:

The meeting went well.

Then the question generated from the adverb *well* is:

How did the meeting go?

2.4 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 [5].

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}*

3 Definitional Question Generation

Definitional questions are questions that have a descriptive answer and can typically be answered only after understanding the whole document. Therefore, the process for generating definitional questions is completely different from the process for generating factoid questions. We generate definitional questions by using the concept of Up-Keys, which are keywords pertaining to the input document [1]. For example, we get Up-Keys like *Lincoln*, *freedom struggle,* etc. from a document about Abraham Lincoln. We already generate Up-Keys in the summarization system used in our QA system [1].

Our procedure for generating definitional questions from an input document is as follows:

1) Extract Up-Keys from the document.
2) Pass each Up-Key to the question word identification module.
3) For each returned question word, generate a question of the type *<Question-word> is <Up-Key>?*

The generated questions demand very descriptive answers that can be expressed only in multiple lines, unlike the answers to factoid questions.

The above strategy produces satisfactory results other than for minor objections in questions relating to places and numbers. For example, the questions generated from the Wikipedia document about Mount Bromo [14] are

1) *What is Mount Bromo?*
2) *What is a volcano?*
3) *Where/Which is Indonesia?*
4) *What is Sand Sea?*

Finding Important Questions. Up-Keys can also be used during factoid question generation. For a given input sentence our factoid question generator produces multiple questions. Not all questions are useful for our QA system. We give more importance to questions which have answers that are Up-Keys since it is highly probable that a query regarding an Up-Key will be given to the QA system.

4 Evaluation

There is no standard way to evaluate the output of a QG system. We use recall and precision, which we define as follows:

$$\text{Precision} = \text{correct} / (\text{correct} + \text{spurious}) \quad (8)$$

$$\text{Recall} = \text{correct} / (\text{correct} + \text{missed}) \quad (9)$$

$$\text{F-score} = 2*\text{Precision}*\text{Recall} / (\text{Precision} + \text{Recall}) \quad (10)$$

where *correct* = the number of questions generated by the QG system and also present in the manually generated questions from a document, *spurious* = the number of questions generated by the QG system but not present in the manually generated questions, and *missed* = the number of questions present in the manually generated questions but not in the questions generated by the QG system. We only evaluate factoid question generation using this method.

We selected twenty sentences from the Brown corpus [7]. We divided the sentences into four categories by length [five in each category], because considerable variation in the quality of question generation output was observed even with slight variation in the sentence length.

We asked five independent judges to generate all possible questions from these sentences. We then ran our factoid question generator on each sentence, and compared the output to the union of the hand-authored questions. Our results are shown in Table 1.

Table 1. Table showing the Precision, Recall and F-Scores obtained by our factoid question generator on 20 sentences

Sentence length in number of words	Precision	Recall	F-Score
0-5	0.57	0.80	0.66
5-10	0.48	0.71	0.57
10-15	0.42	0.66	0.51
15-Above	0.38	0.58	0.46
Average Score	0.46	0.68	0.55

For our factoid question generator, the average precision score was 0.46 and the average recall was 0.68. The average F-score is 0.55. These results are very promising.

5. Discussion and Future Work

We observe that for our system recall is always higher precision. This is because the system always produces more questions than the number of questions generated by the human judges. As it is clear from Table 1, the F-Score decreases as the length of the input sentence increases. This shows the need for improvement of the preprocessing stage, which is responsible for converting long 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.*

In our current work we do not ask questions containing the question word *Why, How much, When, To what extent* etc. To generate such questions, we need an improved named entity recognizer capable of labeling more entity types, including money, dates/times, etc.

Our definitional question generator currently generates questions from all Up-Keys without regard to their importance with respect to the input document. We plan to add modeling of lexical chains, to be able to track the importance of Up-Keys. Finally, we plan to integrate this question generator with a QA system. If an input question matches a question produced by the question generator, then the QA system can simply return the corresponding phrase or sentence rather than searching the document database.

6 Conclusions

In this paper we presented an approach to question generation using parse tree manipulation, named entity recognition, and Up-Keys (significant phrases in a document). We described two question generation methods: one for generating factoid questions, and another for generating definitional questions. We showed how our question generation approach can generate multiple questions from a single input sentence. We demonstrated through an evaluation that our factoid question generation method shows promise, and we discussed our plans to use question generation for question answering.

Acknowledgements. We also would like to thank Dr. Christopher Manning, Associate Professor of Computer Science and Linguistics, Stanford University and the whole Stanford NLP group for their help.

References

[1] Das, R., & Elikkottil, A. (2010). Auto-summarizer to aid a Q/A system. *International Journal of Computer Applications*, 1(1):113–117.
[2] Klein, D., & Manning, C. (2002). Fast exact inference with a factored model for natural language parsing. In *Proceedings of NIPS* (pp. 3-10).
[3] Levy, R., & Andrew, G. (2006). Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of LREC*.
[4] Finkel, J., Grenager, T., & Manning, C. (2005), Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the ACL* (pp. 363-370).
[5] www.eslbee.com/appostives.htm
[6] Fellbaum, C. (Ed.). (1998). *Wordnet : An Electronic Lexical Database*. Cambridge, MA ; London: The MIT Press.
[7] Source of Brown documents http://www.csi.uottawa.ca/tanka/Brown/original/index.html.
[8] Heilman, M., & Smith, N. (2009). *Question generation via overgenerating transformations and ranking*. Technical Report CMU-LTI-09-013, Carnegie Mellon University.
[9] Gates, D. (2008). *Automatically generating reading comprehension look-back strategy questions from expository texts* (Technical Report CMU-LTI-08-011). Pittsburgh: Carnegie Mellon University.
[10] Graesser, A.C., Chipman, P., Haynes, B.C., & Olney, A. (2005). Autotutor: an intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*,48(4), 612–618.
[11] 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).
[12] 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.
[13] Harabagiu, S., Hickl, A., Lehmann, J., & Moldovan, D. (2005). Experiments with interactive question-answering. In *Proceedings of the ACL* (pp. 205-214).
[14] http://en.wikipedia.org/wiki/Mount_Bromo

Extracting Simplified Statements for Factual Question Generation

Michael Heilman and Noah A. Smith

Language Technologies Institute, Carnegie Mellon University
Pittsburgh, PA 15221, USA
{mheilman,nasmith}@cs.cmu.edu

Abstract. We address the problem of automatically generating concise factual questions from linguistically complex sentences in reading materials. We discuss semantic and pragmatic issues that appear in complex sentences, and then we present an algorithm for extracting simplified sentences from appositives, subordinate clauses, and other constructions. We conjecture that our method is useful as a preliminary step in a larger question generation process. Experimental results indicate that our method is more suitable for factual question generation applications than an alternative text compression algorithm.

1 Introduction

This paper addresses question generation (QG) for the purpose of creating reading assessments about the factual information that is present in expository texts such as encyclopedia articles and textbooks. We believe that QG has the potential to help teachers efficiently assess students' acquisition of basic factual knowledge from reading materials, thereby enabling teachers to focus on more complex questions and learning activities. We also believe that research on factual QG can inform research on the generation of more complex reading questions as well as questions for other applications, some of which are described in [17].

Factual questions about reading materials, either from teachers or students, are usually short and targeted at a single piece of information. For example, in the Microsoft Research Question-Answering Corpus,[1] which consists of questions generated by students and excerpts from encyclopedia articles that answer them, the average length of questions is 7.2 words. However, sentences in texts are typically much longer on average because various constructions such as conjunctions, subordinate clauses, and appositives enable writers to convey multiple pieces of information in a single sentence. Consider Ex. 1, which contains a non-restrictive appositive (*the country's paramount leader*), a participial phrase (*returning ...*), and two clauses conjoined by *so*.[2] An important issue in QG is how to generate concise questions from these sorts of complex sentences.

[1] Available at http://research.microsoft.com.

[2] "Subway Blasts Kill Dozens in Moscow" by C. Levy, *New York Times*, downloaded March 29, 2010.

(1) *Prime Minister Vladimir V. Putin, the country's paramount leader, cut short a trip to Siberia, returning to Moscow to oversee the federal response. Mr. Putin built his reputation in part on his success at suppressing terrorism, so the attacks could be considered a challenge to his stature.*

One possible solution is to use techniques from text compression [11, 6]. The goal of compression is to take as input a possibly long and complex sentence, and produce as output a single shortened version that conveys the main piece of information in the input. However, as noted above, sentences often convey multiple pieces of information. In QG, we may want to generate questions not just about the information in the main clause, but also about the information embedded in various nested constructions. For example, from the two sentences in Ex. 1, we might produce the following sentences to allow a QG system to generate a more comprehensive set of questions:[3]

(2) *Prime Minister Vladimir V. Putin is the country's paramount leader.*
(3) *Prime Minister Vladimir V. Putin cut short a trip to Siberia.*
(4) *Prime Minister Vladimir V. Putin returned to Moscow to oversee the federal response.*
(5) *Mr. Putin built his reputation in part on his success at suppressing terrorism.*
(6) *The attacks could be considered a challenge to his stature.*

In this paper, we present a method for extracting multiple, simple, syntactically and semantically correct factual statements from complex sentences. These simple sentences can be readily transformed into questions. There already exist frameworks where questions are generated through transformations of extracted declarative sentences, notably [8, 9]. Our method fits particularly well into sentence-extraction stage of that approach, but could also serve as a preprocessing step for other QG systems.[4] Our method is linguistically motivated by research on semantic and pragmatic phenomena. We present the results of a small-scale study of the quality of the output of our system and compare to a text compression approach [6].

2 Extraction of Textual Entailments

The task of extracting simple sentences from a complex input sentence is essentially the task of generating a particular subset of the possible sentences that a reader would assume to be true after reading the input. That is, we aim to generate a restricted set of textual entailments, using the informal definition of

[3] Ideally, we would also like to resolve anaphora such as *his*. Due to the difficulties of coreference resolution, we leave that to future work.

[4] We consider the set of simplification rules described in [8, 9] to be an early prototype of the more complete system described here. We did not expect a formal experimental comparison to that prototype to be informative.

entailment from the *recognizing* textual entailment task [7, 1]. While it is important to generate outputs that are true given an input sentence, we are not satisfied with outputs that merely preserve meaning. Rather, we seek short sentences such as Exs. 2–6 that are likely to lead to concise questions.

Our method for extracting meaning-preserving, simplified sentences depends on two linguistic phenomena: semantic entailment and presupposition (both of which we subsume into the informal notion of textual entailment).

3 Extraction and Simplification by Semantic Entailment

Semantic entailment is defined as follows: A **semantically entails** B if and only if for every situation in which A is true, B is also true [12].

This section describes how we extract simplifications from complex sentences by removing adjunct modifiers and discourse connectives and by splitting conjunctions of clauses and verb phrases. These transformations preserve the truth conditions of the original input sentence, while producing more concise sentences from which questions can be generated.

3.1 Removing Discourse Markers and Adjunct Modifiers

Many sentences can be simplified by removing certain adjunct modifiers from clauses, verb phrases, and noun phrases. For example, from Ex. 7, we can extract Ex. 8 by removing the discourse marker *however* and the relative clause *which restricted trade with Europe*.

(7) *However, Jefferson did not believe the Embargo Act, which restricted trade with Europe, would hurt the American economy.*
(8) *Jefferson did not believe the Embargo Act would hurt the American economy.*

Ex. 8 is true in all situations where Ex. 7 is true, and is therefore semantically entailed. Discourse markers such as *however* do not affect truth conditions in and of themselves, but rather serve to inform a reader about how the current sentence relates to the preceding discourse. Adjunct modifiers do convey meaning, but again do not affect semantic entailment. Of course, many adjuncts provide useful information that we should preserve for later QG steps. For example, prepositional phrases that identify the locations and times at which events occurred can be the target of *where* and *when* questions, respectively.

Our algorithm (presented in §5) extracts simplified, entailed statements by removing the following adjuncts and discourse markers:

- non-restrictive appositives (e.g., *Jefferson, the third U.S. President, ...*)[5]
- non-restrictive relative clauses (e.g., *Jefferson, who was the third U.S. President, ...*)
- parentheticals (e.g., *Jefferson (1743–1826) ...*)
- participial modifiers of noun phrases (e.g., *Jefferson, being the third U.S. President, ...*)
- verb phrase modifiers offset by commas (e.g., *Jefferson studied many subjects, like the artist Da Vinci. ...*)
- modifiers that precede the subject (e.g., *Being the third U.S. President, Jefferson...*)

We only remove verb phrase modifiers that are offset by commas (e.g., we simplify *When I talked to him, John was busy.* but not *John was busy when I talked to him.*) Whether or not to remove other adjunct verbal modifiers is a challenging question. It may be useful for QG to drop other adjunct modifiers from very long sentences, but deciding which to drop and which to keep is left to future work.

3.2 Splitting Conjunctions

We also split conjunctions of clauses and verb phrases, as in Ex. 5 and Ex. 6 above. In most cases, the conjuncts in these conjunctions are entailed by the original sentence. For example, given *John studied on Monday but went to the park on Tuesday*, both *John studied on Monday* and *John went to the park on Tuesday* are entailed. Exceptions where conjuncts are not entailed include the following: conjunctions with *or* and *nor*, which we do not split; and conjunctions within downward monotone contexts such as negations or non-factive verbs (see [14, Chapter 6] for a discussion of this phenomenon), which we do split (we leave proper handling of this relatively rare case to future work).[6]

4 Extraction by Presupposition

In addition to the strict notion of semantic entailment, the pragmatic phenomenon of presupposition plays an important role in conveying information. The semantically entailed information in complex sentences often covers only a fraction of what readers understand. Consider again Ex. 7 about the Embargo Act. If our algorithm were to only extract semantic entailments as in §3, then it

[5] Appositives and relative clauses can be restrictive or non-restrictive. Restrictive versions resolve referential ambiguities and are not typically offset by commas (e.g., *the man who was tall...*). The non-restrictive ones provide additional information and are offset by commas (e.g., *John, who was tall,...*).

[6] We do not split conjunctions other than those conjoining clauses and verb phrases. In particular, we do not split noun phrases. Leaving conjoined noun phrases is probably advisable for factual QG applications, in particular because of difficult semantic and pragmatic issues involving nouns (e.g., as in *John and Susan were friends*).

would miss possibly useful questions about facts such as Ex. 9 because they are not semantically entailed by Ex. 7 (note how *the Embargo Act would hurt the American economy* is also not entailed).

On the other hand, if an algorithm were to extract from all nested constructions and naïvely copy features such as negation from the main clause, then it would output many false sentences, such as Ex. 10 (from the clause *which restricted trade with Europe*).

(9) *The Embargo Act restricted trade with Europe.*
(10) *The Embargo Act did not restrict trade with Europe.*

As the examples show, information in certain syntactic constructions is not semantically entailed but rather presupposed, or assumed to be true, by the reader [12, Chapter 4]. The meaning conveyed by these constructions is not affected by non-factive verbs like *believe* and *doubt*, negation, modal operators, and other constructions that affect the meaning of the main clause of the sentence.

The phenomenon of presupposition, often subsumed by the term "conventional implicature" [12], can be formally defined as follows: A **presupposes** B if and only if: (a) if A is true, then B is true, and (b) if A is false, then B is true.[7]

Many presuppositions have clear syntactic or lexical associations, or "triggers." These triggers facilitate the extraction of simple statements such as Ex. 2 and Ex. 4 above, which can lead to useful questions. A list of presupposition triggers is provided by [12].[8] The algorithm we present in §5 uses presupposition triggers to extract simplified sentences from the following constructions (in each example provided below, our method will extract *Jefferson was the third U.S. President*):

- non-restrictive appositives (e.g., *Jefferson, the third U.S. President,...*)
- non-restrictive relative clauses (e.g., *Jefferson, who was the third U.S. President,...*)
- participial modifiers (e.g., *Jefferson, being the third U.S. President,...*)
- temporal subordinate clauses (e.g., *Before Jefferson was the third U.S. President...*)

This set of presupposition triggers is a subset of the adjunct modifiers that our method removes when simplifying sentences (§3.1). They cover only subset of the adjunct modifiers because it was not clear how to create reliable and concise extraction rules for all of them. For example, extracting from certain parenthetical phrases, such as in *Jefferson (1743–1826)*, would likely require components to identify time expressions and to generate appropriately formatted output. We leave such extensions to future work.

[7] There are cases where A is neither true nor false, such as *The King of France is tall*. Such instances are often accounted for by abandoning the assumption of bivalence and allowing a third truth value: "neither-true-nor-false." [12, Section 4.2].

[8] The constructions discussed in §3 can be viewed as triggers for semantic entailment.

Algorithm 1 `extractSimplifiedSentences`(t)

$T_{result} \leftarrow \emptyset$
$T_{extracted} \leftarrow \{t\} \cup$ **extract** new sentence trees from t for the following: non-restrictive appositives; non-restrictive relative clauses; subordinate clauses with a subject and finite verb; and participial phrases that modify noun phrases, verb phrases, or clauses.
for all $t \in T_{extracted}$ **do**
  $T_{result} \leftarrow T_{result} \cup$ `extractHelper`(t)
end for
return T_{result}

Algorithm 2 `extractHelper`(t)

$T_{result} = \emptyset$
move any leading prepositional phrases and quotations in t to be the last children of the main verb phrase.
remove the following from t: noun modifiers offset by commas (non-restrictive appositives, non-restrictive relative clauses, parenthetical phrases, participial phrases), verb modifiers offset by commas (subordinate clauses, participial phrases, prepositional phrases), leading modifiers of the main clause (nodes that precede the subject).
if t has `S`, `SBAR`, or `VP` nodes conjoined with a conjunction $c \notin \{or, nor\}$ **then**
  $T_{conjuncts} \leftarrow$ **extract** new sentence trees for each conjunct in the leftmost, topmost set of conjoined `S`, `SBAR`, or `VP` nodes in t.
  for all $t_{conjunct} \in T_{conjuncts}$ **do**
    $T_{result} \leftarrow T_{result} \cup$ `extractHelper`($t_{conjunct}$)
  end for
else if t has a subject and finite main verb **then**
  $T_{result} \leftarrow T_{result} \cup \{t\}$
end if
return T_{result}

5 Extraction Algorithm

This section presents our algorithm for extracting simplified factual statements from complex input sentences. The algorithm operates on standard simplified Penn Treebank [15] phrase structure trees (i.e., without traces, etc.). The primary method `extractSimplifiedSentences`, shown in high-level pseudo-code in Algorithm 1, takes a tree t as input and returns a set of trees T_{result} as output. A helper function, shown in Algorithm 2, recursively splits conjunctions.

As an optional step (enabled for our experiments), the algorithm moves leading prepositional phrase and quote modifiers to the end of the main verb phrase. This transformation does not affect meaning, but it may lead to slightly more natural questions (e.g., *Who was elected President in 1796?* instead of *In 1796, who was elected President?*).

In our implementation, we use the Stanford Parser [10] to find sentence boundaries, tokenize, and parse. We use the `Tregex` tree searching language

[13] to identify the various constructions (e.g., non-restrictive appositives) that our algorithm operates on. After identifying the key nodes with `Tregex`, we manipulate trees using the Stanford Parser's API, which allows for inserting and deleting children, changing labels on tree nodes, etc.[9]

Due to space limitations, we do not include the extraction rules here; they will be listed in a future publication.[10]

6 Evaluation

To evaluate our implementation, we randomly sampled 25 articles from a set of Encyclopedia Britannica[11] articles about cities in the world (provided by [2]).

6.1 Baselines

We compare to two baselines. The first baseline is the Hedge Trimmer (hereafter, HT) algorithm [6], a rule-based text compression algorithm that iteratively performs various simplification operations until an input is shorter than a user-specified target length. We decided to use HT rather than a statistical sentence compression system because [6] found that HT produces more fluent output (that paper compared HT to a hidden Markov model for compression). We implemented HT using `Tregex` rules, and made two modifications to adapt HT for QG: We set the target length to 15, rather than 10. With the target length at 10 in preliminary experiments, HT often led to ungrammatical output and seemed to drop useful information for a QG application. Also, our implementation did not include the rules described in [6] for dropping determiners and temporal clauses (called "low-content units") because these rules are tailored to the headline generation application in [6].

The second baseline (MC-only), is to use our algorithm (§5) but only extract from the main clause (taking left most conjunct when conjunctions are present). This is an intermediate case between HT and our full system: like HT, it only produces only one output per input. Also, note that the outputs will be a subset of those from the full system.

6.2 Experimental Setup

We ran our system and HT on the 25 encyclopedia articles. All systems used the same parser [10]. We set the maximum sentence length for the parser to be

[9] For example, the following rule matches appositive constructions (see [13] for details on the rule syntax): `NP < (NP=noun !$-- NP $+ (/,/ $++ NP|PP=appositive !$CC|CONJP)) >> (ROOT << /^VB.*/=mainverb)`. For each match, the system creates a new sentence of the form, "`noun` *be* `appositive`." The form of *be* depends on the part of speech of the main verb, `mainverb`, and the number and person of `noun`.

[10] Our system, which is coded in Java and includes the extraction rules, is available at http://www.ark.cs.cmu.edu/mheilman/qg-2010-workshop.

[11] Available at http://www.britannica.com.

60 word or punctuation tokens. For the 5.1% of sentences that exceeded this maximum, all systems simply returned the input sentence as output.

We also conducted a small-scale manual evaluation of the quality of the output. We randomly sampled 150 sentences from the encyclopedia articles. For 50 of these, we extracted using HT; for the remaining 100, we extracted using our method (which subsumes MC-only).[12] When the full system generated multiple outputs from an input, we randomly sampled one of them.

We then asked two people not involved with this research to evaluate the 150 outputs for fluency and correctness, on 1–5 scales, following evaluation setups for paraphrase generation [4] and sentence compression [11]. We defined a correct output sentence as one where at least part of the meaning of the input sentence was preserved (i.e., whether the output was true, given the input). The outputs were mixed together, and the raters were blind to which system produced each of them. The Pearson correlation coefficients between the raters' judgments were $r = 0.92$ for fluency and $r = 0.82$ for correctness. For our analyses, we used the averages of the two raters' judgments.

For a rough estimate of how much of the information in the input sentences was included in the outputs, we computed the percentages of input word types (lemmatized with WordNet [16]) included in at least one output, for each system.

6.3 Results

The results of our experiments, presented in Table 1, show that the full system extracts more outputs per input than the two baselines (average 1.63 compared to 1.0). While the main clause baseline constitutes a majority of the output of the full system (58.1%), the other extraction rules contribute a substantial number of additional outputs. For example, 26.5% of outputs involved splitting conjunctions, and 13% were extracted from appositives. Also, the systems do not unnecessarily extract from already simple sentences: 27.9% of the inputs for the full system were unchanged (e.g., *Brussels lies in the central plateaus of Belgium.*).

The outputs from all systems were substantially shorter than the input sentences, which were 23.5 words long on average (not including punctuation). HT outputs were 12.4 words long on average, while outputs from the full system were 13.4 words long on average.[13]

The manual evaluation showed that our system extracts sentences that are, on average, more fluent and correct than those from HT. The mean fluency rating for the full system was 4.75 compared to 3.53 for HT, a difference which is statistically significant (t-test, $p < 0.001$). The mean correctness rating for the full system was 4.67 compared to 3.75 for HT ($p < 0.001$).

[12] We used 2 disjoint sets of sentences so the rater would not compare extractions for the same sentence. We included 100 for the full system rather than 50 in order to get accurate estimates for main clause baseline.

[13] The sentence length values assume the sentence splitting by the Stanford Parser works perfectly. In practice, we observed very few mistakes, mostly on abbreviations ending in periods.

	Input	HT	MC-only	Full
Number of sentences extracted	2952	2952	2952	4820
Sentences per input	1.00	1.00	1.00	1.63
Sentence length	23.5	12.4	15.4	13.4
Input word coverage (%)	100.0	61.4	69.1	87.9
Inputs unchanged (%)	100.0	33.8	27.9	27.9
Fluency	–	3.53	4.72	4.75
Correctness	–	3.75	4.71	4.67
Rated 5 for both Fluency and Correctness (%)	–	44.0	78.7	75.0

Table 1. Various statistics comparing the full system to the baselines and the unmodified input sentences. All statistics are averaged across input sentences.

The full system appears to provide better coverage of the information in the input. 87.9% of input words appeared in at least one output from the full system, compared to 61.4% for HT and 69.1% for the main clause baseline.

In an informal analysis of the errors made by the systems, we observed that the HT baseline is often overly aggressive when compressing sentences: it frequently drops key function words and informative modifiers. For example, from Ex 1, it produces the ungrammatical sentence *Prime Minister Vladimir V. Putin cut short a trip returning oversee the federal response*, and the grammatical but terse sentence *Mr. Putin built his reputation.* In contrast, the mistakes from the MC-only baseline and the full system were mostly the result of parser errors.

7 Related Work on Text Simplification

Numerous approaches to text compression and simplification have been proposed; see [6, 5] for reviews of various techniques. One particularly closely related method is discussed in [3]. That method extracts simple sentences from each verb in the syntactic dependency trees of complex sentences. However, the authors do not provide code or evaluate their system on realistic texts, so it is unclear how robust and effective their approach is (in a small evaluation with one text, 55% of the simplifications extracted by their system were acceptable).

8 Conclusion

In this paper, we presented an algorithm for extracting simplified declarative sentences from syntactially complex sentences. We motivated our extraction approach by identifying semantic and pragmatic issues that are relevant for QG. We evaluated our approach and showed that it is more suitable for extraction of well-formed entailments than a standard text compression baseline.

Our system can be integrated into existing QG frameworks. While we leave formal, extrinsic evaluations within a QG system to future work, we end by reporting that we have incorporated our approach with the QG system described by [8, 9]. The integrated QG system successfully simplifies sentences and transforms them into questions. For example, from Ex. 1 in the introduction, it produces concise questions such as the following:

(11) What did Prime Minister Vladimir V. Putin return to Moscow to oversee?
(12) Who cut short a trip to Siberia?
(13) Who was the country's paramount leader?
(14) Who built his reputation in part on his success at suppressing terrorism?

Acknowledgments. We acknowledge partial support from the Institute of Education Sciences, U.S. Department of Education, through Grant R305B040063 to Carnegie Mellon University; and from the National Science Foundation through a Graduate Research Fellowship for the first author and grant IIS-0915187 to the second author. We thank the anonymous reviewers for their comments.

References

1. Bar-Haim, R., Dagan, I., Greental, I., Szpektor, I., Friedman, M.: Semantic inference at the lexical-syntactic level for textual entailment recognition. In: Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing (2007)
2. Barzilay, R., Elhadad, N.: Sentence alignment for monolingual comparable corpora. In: Proc. of EMNLP (2003)
3. Beigman Klebanov, B., Knight, K., Marcu, D.: Text simplification for information seeking applications. On the Move to Meaningful Internet Systems (2004)
4. Callison-Burch, C.: Paraphrasing and Translation. Ph.D. thesis, University of Edinburgh (2007)
5. Clarke, J.: Global Inference for Sentence Compression: An Integer Linear Programming Approach. Ph.D. thesis, University of Edinburgh (2008)
6. Dorr, B., Zajic, D.: Hedge Trimmer: A parse-and-trim approach to headline generation. In: Proc. of Workshop on Automatic Summarization (2003)
7. Giampiccolo, D., Magnini, B., Dagan, I., Dolan, B. (eds.): The third pascal recognizing textual entailment challenge (2007)
8. Heilman, M., Smith, N.A.: Question generation via overgenerating transformations and ranking. Tech. Rep. CMU-LTI-09-013, Language Technologies Institute, Carnegie Mellon University (2009)
9. Heilman, M., Smith, N.A.: Good question! statistical ranking for question generation. In: Proc. of NAACL-HLT (2010)
10. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: Advances in NIPS 15 (2003)
11. Knight, K., Marcu, D.: Statistics-based summarization - step one: Sentence compression. In: Proc. of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence (2000)
12. Levinson, S.C.: Pragmatics. Cambridge University Press, Cambridge, MA (1983)
13. Levy, R., Andrew, G.: Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In: Proc. of LREC (2006)
14. MacCartney, B.: Natural language inference. Ph.D. thesis, Stanford University (2009)
15. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of English: The Penn Treebank. Computational Linguistics 19 (1993)
16. Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J.: WordNet: An on-line lexical database. International Journal of Lexicography 3(4) (1990)
17. Rus, V., Graessar, A. (eds.): The Question Generation Shared Task and Evaluation Challenge (2009), available at http://www.questiongeneration.org/

The Effects of Cognitive Disequilibrium on Question Generation While Interacting with AutoTutor

Jeremiah Sullins[1,1], Katarina Tran[1], Savannah Ewing[1], Natasha Velaga[1], Art Graesser[1],

[1] University of Memphis, Department of Psychology, 400 Innovation Drive, 38152, Memphis, TN, USA
{jsullins@memphis.edu, khtran@memphis.edu, smwlson@memphis.edu, ngundapaneni@gmail.com, art-graesser@mail.psyc.memphis.edu}

Abstract. The purpose of this study was to test the effects of cognitive disequilibrium on student question generation while interacting with an intelligent tutoring system. Some students were placed in a state of cognitive disequilibrium while they interacted with AutoTutor on topics of computer literacy. During the course of the study, a confederate was present to answer any questions that the participant may have had. Preliminary results revealed a significant difference in the amount of questions asked between participants using the two different versions of AutoTutor. Additionally, significant correlations were found between number of questions asked and tests of individual differences.

Keywords: Question Asking, Emotions, Confusion, Cognitive Disequilibrium, Intelligent Tutoring Systems

1 Introduction

Question generation has received a great deal of attention in recent years from researchers in the field of computer science and psychology [1], [2], [3]. Question generation is believed to play a crucial role in a variety of cognitive faculties, such as comprehension [4], [5] and reasoning [6], [7]. Asking good questions has been shown to lead to improved memory and comprehension of material among school children. This research suggests that learning how to ask good questions should be taught at an early age. If this vision could be realized, then students would possess the training to mature into curious question generators.

Sadly, it is well documented that this "ideal" student scenario is not the case. It has been shown that students are inefficient at monitoring their own knowledge deficits and that their question generation is infrequent and unsophisticated (e.g., [8], [9], [10], [11]). [10] pointed out that an individual student asks approximately 1 question

in 7 hours of class time (around 1 question per day). Most of these questions are not good questions, so the quality is also disappointing.

However, imagine a scenario in which a student is given a learning task where obstacles are in place to challenge the student. During the learning task the student may encounter concepts that are unfamiliar to them or ideas that are contradictory. This unfamiliar or contradictory content may place the student in a state of cognitive disequilibrium. Cognitive disequilibrium is a situation in which learners are presented with obstacles to goals, anomalous events, contradictions, discrepancies, and/or obvious gaps in knowledge within the learner's range of knowledge [12], [13], [14], [15]. Cognitive disequilibrium occurs when a learner is forced to handle an obstacle that prevents them from achieving a goal. This disparity between student knowledge and outside events/information creates a state of confusion and requires extra processing in order to integrate the two sources. Students that are in a state of cognitive disequilibrium are potentially in a perfect scenario to significantly increase the amount of questions asked in order to restore cognitive equilibrium.

1.1 AutoTutor

The tutoring system that was used in this experiment is AutoTutor, which is a fully automated tutor that holds conversations with learners in natural language and simulates the dialogue moves of expert human tutors. AutoTutor adheres to constructivist theories of pedagogy by guiding students to actively construct answers to difficult questions, as opposed to merely presenting the correct information to the students. For a detailed discussion of AutoTutor's dialogue mechanisms and strategies, see [16].

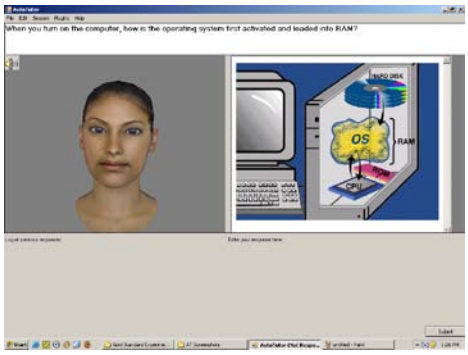


Figure 1. Example screenshot of AutoTutor for computer literacy.

1.2 Confusion and Learning

In the past, researchers thought that emotions were merely a motivational factor and had no influence on complex learning. However, in recent years the link between

emotions and learning has received a growing interest in education, psychology, computational linguistics, and artificial intelligence [17], [18], [19], [20], [21], [22], [23], [24].

A study that directly tested the relationship between emotions and learning was conducted by [25]. In this study five participants interacted with an intelligent tutoring system, called AutoTutor, for 90 minutes. The students were asked to emote-aloud after they were trained on the relevant emotions, which consisted of anger, boredom, confusion, contempt, curiosity, disgust, eureka, and frustration. Results revealed a small number of occurrences for anger (n = 17), contempt (n = 8), curiosity (n = 1), and disgust (n = 5) so these emotions were removed from subsequent analyses. In contrast, sufficient frequencies of emotions were available for boredom (n = 40), confusion (n = 53), delight (n = 28), and frustration (n = 49). Analyses revealed significant correlations between AutoTutor's dialog and three emotions: confusion, delight, and frustration.

These results suggest that not only are emotions present during learning but that emotions are linked to cognition. More specifically, these results provide evidence that emotions (i.e., confusion, frustration and boredom) are a crucial component during learning. When students experience confusion, they are placed in a state of inquiry which in turn drives questions. Furthermore, these studies suggest cognitive disequilibrium may be manifested by confusion. When students are encountering information that they may not know or contradicts what they already know it is placing them in a state of cognitive disequilibrium which is manifested on the face through the display of confusion.

1.3 Emotion Detection and Cognitive Disequilibrium

Research has revealed that confusion is a dominant factor when studying learner emotions while interacting with intelligent tutoring systems. In numerous studies, a significant positive correlation has been revealed between learning and confusion [26], [27], [19], [25]. The relation existing between confusion and learning provides additional support for the importance of cognitive disequilibrium [12], [28]. During students' interactions with an intelligent tutoring system, they may encounter new incoming information which reveals knowledge gaps or discrepancies with existing prior knowledge. Such obstacles may place the student in a state of inquiry where they will ask questions until equilibrium is restored.

In one example, [29] investigated two methods of identifying affective states while students interacted with an intelligent tutoring system, called AutoTutor. During their interaction with AutoTutor, learners were told to emote-aloud. During this study, participants were asked to simply state the affective state that they were feeling while they were interacting with the tutoring system. Two trained judges used the FACS to independently rate the action units that occurred 3 seconds prior to each learner utterance and during the utterance. Analyses were able to determine significant relationships with action units for frustration, boredom, and confusion. More specifically, it appeared that action units 1 (inner brow raiser), 2 (outer brow raiser), and 14 (dimpler) were associated with boredom. Confusion displayed action units 4 (lowered brow), 7 (tightened lids), and 12 (lip corner puller). Boredom displayed a

significant association with action unit 43 (eye closure). These results provide evidence that specific facial action units do occur with certain emotions.

Similarly, a study conducted by [30] used different methodologies to explore emotions while students interacted with AutoTutor. More specifically, they investigated facial features to detect the emotions that accompany deep-level learning of conceptual material. Participants interacted with AutoTutor for 32 minutes while the system recorded the participant's face, body posture, and the computer screen. The participant's session was then observed and scored by four different judges (the learner, a peer, and two trained judges). Following the previous research on emotions related to learning, the emotions assessed in this study were boredom, confusion, flow, frustration, delight, neutral, and surprise. Raters watched the learners' sessions and were told to rate any emotion they saw every 20 seconds (mandatory judgments). Raters were also instructed to rate any emotions they observed within the 20 second interval (voluntary judgments). Results revealed that confusion was manifested by action unit 4 (lowering of the brow) and action unit 7 (tightening of the eyelids). The results from the two previously mentioned studies provide a methodological framework to begin to explore the possible relation between action units, emotions and student questions.

2 Current Experiment

2.1 Participants

Participants consisted of 48 undergraduate students. Participants were recruited from the Psychology Subject Pool at the University of Memphis and received course credit in return for their participation.

2.2 Procedure

Students were trained using one of two versions of AutoTutor which were designed to test the impact of cognitive disequilibrium on questions and emotions. The experiment consisted of three different phases: a pretest phase, a training phase, and a posttest phase.

Pretest Phase. Participants were tested individually during a two-hour session. First, participants completed a demographic questionnaire, the Big Five Personality Test and a 24 item domain knowledge questionnaire. Versions A and B of the domain knowledge questionnaire were counterbalanced with respect to the pre and posttest.

Training Phase. Participants were instructed to take a seat at the computer console where they were introduced to the confederate. The experimenter instructed both the confederate and the participant on their roles for the experiment. The participant was always assigned the role of "primary communicator". They were told that it is their job to collaborate with the confederate, and to enter all input using the keyboard into AutoTutor. The confederate was always be assigned to the role of "secondary

communicator". They were told that it is their job to collaborate with the participant and assist in any way necessary (e.g., answering questions). On the basis of random assignment, learners interacted with either a version of AutoTutor that provided false feedback and false information (*cognitive disequilibrium AutoTutor)* or a version of AutoTutor that gave all correct feedback and information (*regular AutoTutor)*. Students' questions and facial action units were recorded throughout the duration of the learning session. Learners' facial action units were captured using a standard webcam. The webcam was positioned in a way to capture each participant's face at all times. Additionally, all dialogs between the participant and the confederate (including student questions) were captured using a standard desktop microphone. Following the completion of the study, the experimenter reviewed all video and audio recordings to code for all student questions and any facial action units that arise. A second trained judge also coded questions and facial action units in order to assess inter-rater reliability.

Posttest Phase. During the posttest phase participants were first asked to complete an agent persona questionnaire which assessed AutoTutor on a number of different measures (e.g., feedback and usability). Participants then completed a second test assessing their knowledge of computer literacy (counterbalanced with the first assessment). Following the knowledge assessment, participants completed the Motivated Strategies Learning Questionnaire. Finally, participants were asked to complete a confederate perception questionnaire which assessed the confederate on a number of different aspects (e.g., likeability and believability).

2.3 FACS and Question Coding

Several properties of the session were observed and coded. First, a trained judge observed the videos and determined when a question has been addressed to the confederate. The judge rated that question as either deep or shallow based on the [10] question taxonomy. Second, the judge observed the videos and coded each individual's action units three seconds before a question is asked, during a question and three seconds after a question has been asked. A second judge rated all facial action units and student questions in order to obtain inter-rater reliability.

3 Preliminary Results

Although all of the data analysis has not yet been completed, preliminary analyses do reveal some interesting results. For example, analysis revealed a significant difference in the amount of questions asked between the participants in the regular AutoTutor condition (M=61.5) versus the cognitive disequilibrium AutoTutor condition (M=32.1) $t(20) = 2.479$, $p < .05$. Additionally, a significant positive correlation was discovered between the amount of questions asked during the tutoring session and various measures on the Motivated Strategies Learning Questionnaire. More specifically, a significant positive correlation was found between number of questions asked and intrinsic goal orientation $r(22) = .599$, $p < .01$, task value $r(22) =$

.476, $p < .05$, self efficacy for learning and performance $r(22) = .457$, $p < .05$, elaboration strategies $r(22) = .48$, $p < .05$, and effort regulation $r(22) = .479$, $p < .05$. A marginally significant relationship was found between number of questions asked and help seeking behavior $r(22) = .382$, $p = .08$. Significant positive correlations were also found between number of questions asked and the Big Five Personality Test. More specifically, a significant positive correlation was found between number of questions asked and conscientiousness $r(22) = .431$, $p < .05$, and extraversion $r(22) = .492$, $p < .05$.

It was also discovered that the participants in the two conditions significantly differed in how they viewed certain aspects of AutoTutor. For example, participants using the regular version of AutoTutor rated AutoTutor significantly higher on the following questions: "While covering the material, I tried to make everything fit together" $t(46) = 2.039$, $p < .05$, "AutoTutor's emotions were natural" $t(46) = 2.506$, $p < .05$, and "The feedback from AutoTutor was appropriate with respect to my progress" $t(46) = 2.706$, $p < .05$.

Participants also significantly differed on how they viewed certain characteristics of the confederates. More specifically, participants using the regular version of AutoTutor rated the confederates significantly higher on the following items: "I found the confederate to be likeable" $t(46) = 2.208$, $p < .05$, "I felt that the confederate was able to answer my questions during the study" $t(46) = 2.298$, $p < .05$, "I felt that the confederate was knowledgeable" $t(46) = 2.255$, $p < .05$.

3 Discussion

These results shed some light on the relationship between cognitive disequilibrium and student question asking. Results revealed that participants using the regular version of AutoTutor asked significantly more questions during the session than did the participants using the cognitive disequilibrium version of AutoTutor. This is an unexpected finding that is contradictory to what the cognitive disequilibrium literature would suggest. One possible explanation for this is in the nature of the answers given by the confederates to the questions asked by the participants. The confederates were told to answer each question to the best of their ability. Because of this, participants using the regular version of AutoTutor were more likely to receive positive feedback from AutoTutor based on the answers they received from the confederate. However, in the cognitive disequilibrium condition, the confederates still answered to the best of their ability but due to the false feedback and false information from AutoTutor the confederates' answer might not have been the correct answer. Additionally, it may be that the participants did not have enough interest in the topic of computer literacy to notice or care about any of the false information or false feedback. Future research exploring cognitive disequilibrium may want to cover more controversial topics that participants have an invested interest in (e.g., global warming or euthanasia).

Future analyses for the current study will include a time series analysis (5 minute intervals) to determine exactly when questions are asked over the entire session between the two conditions. Additionally, in order to determine if the participants in the regular AutoTutor condition asked not only more questions but "better" questions,

all questions will be coded and scored on quality based on a question taxonomy. Lastly, all action units (based on the facial action coding system) will be coded three seconds prior to every question, during every question, and three seconds after every question in order to determine exactly when cognitive disequilibrium occurs in relation to a student generated question. In other words, it could be that participants are receiving information from AutoTutor which in turn triggers cognitive disequilibrium (measure by action units) which causes a question to be generated. On the other hand, participants may ask a question which in turns reveals a misconception leading cognitive disequilibrium to occur after a question has been asked.

References

1. Rus, V. & Graesser, A.C. (Eds.). (2009). The Question Generation Shared Task and Evaluation Challenge. ISBN:978-0-615-27428-7.
2. Sullins, J., & McNamara, D.S. (2009). iSTART question training module: Training students efficient questioning skills. Presented at the 2nd Workshop on Question Generation, Brighton, United Kingdom.
3. Graesser, A., Ozuru, Y., & Sullins, J. (2009). What is a good question? In M. G. McKeown & L. Kucan (Eds.), *Threads of coherence in research on the development of reading ability* (pp. 112-141). NY: Guilford.
4. Collins, A., Brown, J. S., & Larkin, K. M. (1980). Inference in text understanding. In R. J. Spiro, B. C. Bruce, & W. F. Brewer (Eds.) *Theoretical issues in reading comprehension* (pp. 385-407). Hillsdale, NJ: Erlbaum.
5. Graesser, A., Singer, M., & Trabasso, T. (1994). Constructing inferences during narrative text comprehension. *Psychological Review, 3*, 371-395.
6. Graesser, A. C., Baggett, W., & Williams, K. (1996). Question-driven explanatory reasoning. *Applied Cognitive Psychology, 10*, 17-32.
7. Sternberg, R. J. (1987). Questioning and intelligence. *Question Exchange, 1*, 11-13.
8. Baker, L. (1979). Comprehension monitoring: Identifying and coping with text confusions. *Journal of Reading Behavior, 11,* 363-374.
9. Dillon, J. T. (1988). *Questioning and teaching: A manual of practice*. New York: Teachers College Press.
10. Graesser, A. C, & Person, N. (1994). Question asking during tutoring. *American Educational Research Journal, 31*, 104-137.
11. Van der Meij, H. (1988). Constraints on question asking in classrooms. *Journal of Educational Psychology, 80*, 401-405.
12. Otero, J., & Graesser, A.C. (2001). PREG: Elements of a model of question asking. *Cognition & Instruction, 19,* 143-175.
13. Graesser, A.C., Lu, S., Olde, B.A., Cooper-Pye, E., & Whitten, S. (2005). Question asking and eye tracking during cognitive disequilibrium: Comprehending illustrated texts on devices when the devices break down. *Memory and Cognition, 33,* 1235-1247.
14. Wisher, R.A., & Graesser, A.C. (2005). Question asking in advanced distributed learning environments. In S.M. Fiore and E. Salas (Eds.), *Toward a science of distributed learning and training.* Washington, D.C.: American Psychological Association.
15. Graesser, A. C., & McMahen, C. L. (1993). Anomalous information triggers questions when adults solve problems and comprehend stories. *Journal of Educational Psychology, 85,* 136-151.

16. Graesser, A.C., Chipman, P., Haynes, B.C., & Olney, A. (2005). AutoTutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions in Education,* 48, 612-618.
17. Breazeal, C. (2003). *Designing sociable robots* .Cambridge: MIT Press.
18. Conati C. (2002). Probabilistic assessment of user's emotions in educational games. *Journal of Applied Artificial Intelligence, 16*, 555-575.
19. Craig, S.D., Graesser, A. C., Sullins, J., & Gholson, B. (2004a). Affect and learning: An exploratory look into the role of affect in learning. *Journal of Educational Media, 29,* 241-250.
20. Kort, B., Reilly, R., & Picard, R. (2001). An affective model of interplay between emotions and learning: Reengineering educational pedagogy—building a learning companion. In T. Okamoto, R. Hartley, Kinshuk, & J. P. Klus (Eds.), *Proceedings IEEE International Conference on Advanced Learning Technology: Issues, Achievements and Challenges* (pp. 43-48). Madison, Wisconsin: IEEE Computer Society.
21. Lepper, M. R., & Woolverton, M. (2002). The wisdom of practice: Lessons learned from the study of highly effective tutors. In J. Aronson (Ed.), *Improving academic achievement: Impact of psychological factors on education* (pp. 135-158). Orlando, FL: Academic Press.
22. Litman, D. J., & Forbes-Riley, K. (2004). Predicting student emotions in computer-human tutoring dialogues. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (pp. 352-359)*. East Stroudsburg, PA: Association for Computational Linguistics.
23. De Vicente, A., & Pain, H. (2002). Informing the detection of students' motivational state : An empirical study. In S.A. Cerri, G. Gouarderes, and F. Paraguacu (Eds.), *Intelligent tutoring systems 2002*. Berlin, Germany: Springer.
24. Picard, R. W. (1997). *Affective computing*. Cambridge: MIT Press.
25. Graesser, A.C., D'Mello, S.K., Craig, S.D., Witherspoon, A., Sullins, J., McDaniel, B., & Gholson, B. (2008). The relationship between affect states and dialogue patterns during interactions with AutoTutor. *Journal of Interactive Learning Research, 19*, 293-312.
26. Graesser, A.C., McDaniel, B., Chipman, P., Witherspoon, A., DÕMello, S.K., & Gholson, B. (2006). Detection of Emotions during Learning with AutoTutor. *Proceedings of the 28th Annual Meetings of the Cognitive Science Society*. Mahwah, NJ: Erlbaum.
27. Graesser, A. C., D'Mello, S. K., Chipman, P., King, B., and McDaniel, B. (2007). Exploring Relationships Between Affect and Learning with AutoTutor. *Supplementary Proceedings of the 13th International Conference on Artificial Intelligence in Education (AIED 2007)*, (pp 16-23).
28. Graesser, A.C., & Olde, B.A. (2003). How does one know whether a person understands a device? The quality of the questions the person asks when the device breaks down. *Journal of Educational Psychology, 95,* 524-536.
29. Craig, S. D., D'Mello, S., Witherspoon, A., & Graesser, A. (2008). Emote-aloud during learning with AutoTutor: Applying the facial action coding system to affective states during learning. *Cognition and Emotion, 22*, 777 – 788.
30. McDaniel, B. T., D'Mello, S. K., King, B. G., Chipman, P., Tapp, K., & Graesser, A. C. (2007). Facial Features for Affective State Detection in Learning Environments. In D. S. McNamara & J. G. Trafton (Eds.), *Proceedings of the 29th Annual Cognitive Science Society* (pp. 467-472). Austin, TX: Cognitive Science Society.

Question Generation in the CODA project

Paul Piwek and Svetlana Stoyanchev

Centre for Research in Computing,
The Open University, Milton Keynes, UK
{p.piwek,s.stoyanchev}@open.ac.uk

Abstract. In the ongoing CODA project, we are developing a system for automatically converting monologue into dialogue. The dialogue is generated in a two-step approach. Firstly, snippets of input monologue are mapped to dialogue act sequences. Secondly, these sequences are verbalized. The conversion relies partly on analysing input monologue in terms of its discourse relations. This short paper briefly describes the approach to the first step in CODA. This approach involves the use of a parallel corpus of monologues and dialogues to learn mappings from monologue to dialogue acts. Here, we focus on dialogue acts that involve question asking.

Key words: dialogue generation, parallel monologue/dialogue corpus, language generation, question generation

1 Introduction

CODA[1] is a 2-year project that started in July 2009. The overall aim of CODA is to develop a system for automatically generating dialogue from monologue. This paper provides a brief overview of the work so far, concentrating on the generation of questions (which is a part of generating dialogue).

Dialogue as a means of information presentation goes back at least as far as the ancient Greeks – Plato conveyed his philosophy through fictitious conversations between Socrates and his contemporaries. In this kind of dialogue, henceforth *expository dialogue*, the main purpose is to inform the reader or audience; the information the dialogue partners convey to one another is subservient to this purpose.

Craig et al. [2] found that presenting tutorials as dialogues that are viewed by the student has advantages over presenting the same information as monologue; the benefits of dialogue include stimulating students to write more in a free recall test and to ask twice as many deep-level reasoning questions in a transfer task. Further benefits of dialogue presentation are summarized in [8].

Despite the evidence that in some situations dialogue can be more effective for presenting information than monologue, for the foreseeable future, *text* in monologue form is likely to remain the most common medium for the production of information content, e.g., in the form of books, articles, web pages, and leaflets.

[1] COherent Dialogue Automatically generated from text

Writing monologue is practised by most authors while writing in the form of a dialogue is a less common practice. The aim of CODA is to allow existing monologue content to be presented as dialogue. In this project we develop the theory and technology for automatic transformation of text in monologue form to expository dialogue, specifically dialogues between a 'layman' (e.g., patient or student) and 'expert' (e.g., doctor or tutor) character. Our goal is to present monologue in the form of a dialogue to observers with the purpose of informing them. It is different from the task of interactive tutoring dialogue systems [4, 3, 12] where the system plays a role of a tutor and a user is one of the participants in the dialogue.

The approach to dialogue generation taken in CODA differs from most of the existing work on expository dialogue generation, e.g., for Embodied Conversational Agents (see [11]), in that it starts from text rather than information stored in a database or knowledge representation. The work also differs from the pilot study we carried out on text to dialogue generation (see [7]), in that it aims to use mapping rules that have an empirical grounding. In particular, we have constructed a corpus by translating professionally authored dialogues into monologues. The CODA parallel corpus[2] consists of dialogue segments aligned with monologue snippets expressing the same content. A recent study showed that human-authored monologue-to-dialogue mapping rules can account for only a small percentage (about 13%) of the dialogue structures that are found in professionally authored dialogues [5].

In the next section, we introduce the overall architecture of the CODA system and the corpus from which monologue to dialogue mapping rules were extracted. In Section 3, we describe our findings relating to the generation of questions. The paper ends with a conclusions section.

2 Generating Dialogue in CODA

For the CODA system, see Figure 1, the input is a monologue. Off-the-shelf technologies are used for parsing its syntactic and discourse structure. The annotated text is processed in two steps. Firstly, the text is mapped to a sequence of dialogue acts. In a second step, these acts are verbalized. The resulting dialogue can be rendered either as text (via 6a and 7a) or as a video of computer-animated agents (6b and 7b), with MPML3D [9] as the interface language to the agents.

The repository of rules for mapping discourse relations in text to dialogue acts is automatically extracted (for more detail see [8]) from the CODA corpus. The corpus currently consists of 800 turns from samples of dialogues written by acclaimed authors such as Mark Twain. For each dialogue sample, a monologue was written expressing the same content. Dialogue spans were then aligned with snippets of monologue. The dialogue side was annotated with dialogue acts and the monologue side with discourse relations. The corpus construction is described in [10] together with interannotor agreement information.

[2] The corpus will be released at `http://computing.open.ac.uk/coda/` in June 2010.

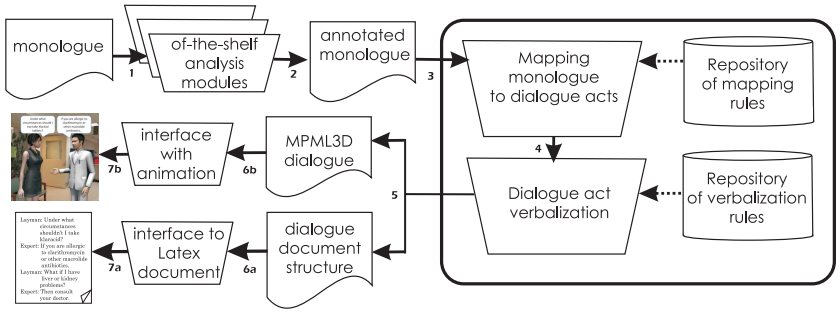


Fig. 1. CODA Architecture for transforming text to dialogue

3 Questions in Dialogue

A dialogue is a sequence of dialogue acts including questions, answers, and responses signalling agreement or contradiction. The CODA corpus aligns dialogue sequences written by professional writers with monologue segments. In our corpus we use three types of question dialogue acts (presented together with examples and frequency from a 259 turn sample of the CODA corpus):

Question type	*Example*	*Frequency*
Yes/No	'Can it force a right-principled man to do a wrong thing?'	173 instances
Complex	'How are you going to make that out, when the lower animals have no mental quality but instinct, while man possesses reason?'	65 instances
Factoid	'How many times did you try the experiment?'	18 instances

Table 1. Types of question asking dialogue acts in the CODA corpus and their frequencies in a 259 turn sample

Dialogue (including questions) generation in CODA project involves two steps:

1. determining the dialogue act sequence; and
2. verbalizing the dialogue sequence.

Our mapping rules from text to dialogue acts address the first step of dialogue generation: determining which dialogue acts may be used to express a particular bit of input monologue in dialogue generation. This task is closely related to the Nielsen's Question Type Determination task [6]. Nielsen's task, one of three tasks central to Question Generation, is about determining the most appropriate type of question which can be asked for an input text. Our task, in addition to question type determination, also involves identifying dialogue acts for the responses to the questions, such as *Agree* or *Contradict*. To illustrate this point, let us look at a specific rule that was automatically extracted from the CODA corpus:

$$\textsc{attribution}(P,Q) \quad \Longrightarrow \quad \begin{array}{l}\text{Expert: yes/no InfoRequest}(Q),\\ \text{Layman: Resp-Answer-Yes}(P)\end{array}$$

This rule was extracted from the following dialogue fragment in the CODA corpus (out of Twain's *What is man?*):

> Expert: He felt well?
> Layman: One cannot doubt it.

In the corpus, the dialogue fragment is aligned with the following monologue segment:

> [One cannot doubt]$_1$ [that he felt well]$_2$'

The monologue segment is accompanied by an annotation with the discourse relation Attribution(1,2).

70% of the rules that we automatically extracted are based on a discourse relation in the monologue. In other words, most sequences of dialogue acts (forming a coherent dialogue fragment)[3] were aligned with text held together by discourse relations. However, not all rules involve discourse relations. Some monologue snippets map to several dialogue acts, even though they do not include a discourse relation. For example, the monologue snippet

> However, on his way home his mind was in a state of joy which only the self-sacrificer knows.

maps to the following dialogue fragment:

> Expert: What was his state of mind on his way home?
> Layman: It was a state of joy which only the self-sacrificer knows.

In this example, the *what* question is based on the semantics of the statement, not on discourse structure.

Both for rules with and without discourse relations the majority (74% and 78%, respectively) mapped to dialogue act sequences that included a question.

The second task of dialogue sequence verbalization is achieved with dialogue move verbalization rules. Such rules for verbalizing questions have been constructed in previous work on question generation (e.g., Wyse and Piwek [13]). In our dialogue generation task, we reuse previously constructed question generation rules as well as rules that are harvested from the CODA corpus. The corpus provides sentence-question pairs from which we aim to extract, in the first instance, manually further dialogue move verbalization rules. We aim also aim to experiment with methods from the paraphrasing literature (e.g., Barzilay and McKeown [1]) for automatically deriving the dialogue act verbalization rules.

[3] I.e., translatable to a sentence or a paragraph that conveys a complete point.

4 Concluding Remarks

This paper described the ongoing CODA project and the generation of questions as an integral part of dialogue generation from monologue. We have provided some information on questions found in the parallel CODA corpus and the relation of these questions to the monologue with which they are aligned in the corpus.

We are still early on in the project, but have already several outcomes. These include a parallel corpus of monologues and dialogues (and a dedicated tool for creating such corpora) and a repository of high-level rules for mapping discourse relations to dialogue act sequences. We're currently working on verbalization of the dialogue acts. For the second year of the project an evaluation study is planned. This will focus on evaluating the fluency and coherence of automatically generated dialogue, and also whether the generated dialogues preserve the information in the input monologues.

Finally, with regards to the CODA corpus, we believe that it can be a useful resource not just for monologue-to-dialogue generation, but also more generally for question generation.

Acknowledgments. We would like to thank the three anonymous reviewers for QG2010 for their helpful comments. The research reported in this paper is funded by the UK Engineering and Physical Sciences Research Council under grant EP/G/020981/1.

References

1. R. Barzilay and K. McKeown. Extracting paraphrases from a parallel corpus. In *Proc. of ACL/EACL*, Toulouse, 2001.
2. S. Craig, B. Gholson, M. Ventura, A. Graesser, and the Tutoring Research Group. Overhearing dialogues and monologues in virtual tutoring sessions. *International Journal of Artificial Intelligence in Education*, 11:242–253, 2000.
3. Martha W. Evens and Joel A. Michael. *One-on-one Tutoring by Humans and Machines*. Mahwah, NJ: Lawrence Erlbaum Associates, 2006.
4. Arthur C. Graesser, S. Lu, G.T. Jackson, H. Mitchell, M. Ventura, A. Olney, and M.M. Louwerse. AutoTutor: A tutor with dialogue in natural language. *Behavioral Research Methods, Instruments, and Computers*, 36:180–193, 2004.
5. N. Hastings. Towards Transforming Monologues into Dialogues Automatically. Master's thesis, Computing Department, The Open University, March 2010.
6. R. Nielsen. Question generation: Proposed challenge tasks and their evaluation. In V. Rus and A. Graesser, editors, *Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge*, Arlington, Virginia, September 2008.
7. P. Piwek, H. Hernault, H. Prendinger, and M. Ishizuka. T2D: Generating Dialogues between Virtual Agents Automatically from Text. In *Intelligent Virtual Agents*, LNAI 4722, pages 161–174. Springer Verlag, 2007.

8. P. Piwek and S. Stoyanchev. Generating Expository Dialogue from Monologue: Motivation, Corpus and Preliminary Rules. In *Procs of NAACL-HLT 2010*, Los Angeles, June 2010.
9. H. Prendinger, S. Ullrich, A. Nakasone, and M. Ishizuka. MPML3D: Scripting Agents for the 3D Internet. *IEEE Trans. on Visualization and Computer Graphics*, 2010.
10. S. Stoyanchev and P. Piwek. Constructing the CODA corpus. In *Procs of LREC 2010*, Malta, May 2010.
11. K. van Deemter, B. Krenn, P. Piwek, M. Klesen, M. Schröder, and S. Baumann. Fully generated scripted dialogue for embodied agents. *Artificial Intelligence Journal*, 172(10):1219–1244, 2008.
12. Kurt VanLehn, Arthur C. Graesser, G. Tanner Jackson, Pamela W. Jordan, Andrew Olney, and Carolyn P. Rosé. When are tutorial dialogues more effective than reading? *Cognitive Science*, 31(1):3–62, 2007.
13. B. Wyse and P. Piwek. Generating questions from openlearn study units. In V. Rus and J. Lester, editors, *Proceedings of the 2nd Workshop on Question Generation, AIED 2009 Workshop Proceedings*, pages 66–73, 2009.

What's next? Target Concept Identification and Sequencing

Lee Becker[1], Rodney D. Nielsen[1,2], Ifeyinwa Okoye[1], Tamara Sumner[1], and Wayne H. Ward[1,2]

[1] Center for Computational Language and Education Research,
University of Colorado at Boulder, Boulder, CO, 80309 USA
{lee.becker, rodney.nielsen, ifeyinwa.okoye, tamara.sumner}@colorado.edu,
[2] Boulder Language Technologies, Boulder, CO 80301 USA
wward@bltek.com

Abstract. We define the first and arguably most important step in many applications of question generation, the identification and sequencing of the key concepts in a knowledge source (i.e., the concepts that are question worthy). We describe component and baseline systems developed to address this task and present their evaluation. We demonstrate that the subtasks are feasible and challenge the question generation community at large to undertake this task as a critical step in the overall process of question generation.

1 Introduction

There has been scattered work in Question Generation (QG) going back at least as far as ELIZA, but a growing multidisciplinary group is gaining momentum on the heels of the QG workshops and challenge [15]. This group and the challenge will have a significant effect on the future of question generation and, hence, it is critical that the community take a broad perspective and not view QG as being primarily surface form realization or roughly a syntactic transformation that converts a, typically, declarative form to an interrogative form. We must consider all components of the overall QG framework and ensure we do not forget the importance of focusing on question-worthy concepts.

Most QG applications can be conceived of as dialogue systems, where the question generated will depend not only on the text or database from which the associated knowledge is drawn, but also the context of all previous interactions. Even an educational *assessment* application optimally should adapt to the student's performance on previous questions [19]. Given a dialogue context, QG can be viewed as a three-part process [9]. In the first task, *Target Concept Identification*, the topic from which a question is to be generated is determined. In the second (potentially concurrent) task, *Question Type Determination*, a decision is made about the type of question to be asked. In the final task, *Question Realization*, the surface form of the question is created based on the prior steps.

Target Concept Identification is the identification of the most appropriate concept, out of the limitless number of concepts related to the current dialogue

context, from which to construct the next question in a dialogue. This involves deciding which concepts in general are worth discussing, what their relative merit is, and how they depend on one another. This task depends both on the domain and the social role of the intelligent system – in a tutoring system, one strategy might be to intentionally drive students toward an area of limited understanding and difficult-to-answer questions; whereas, medical assistance applications will focus on questions they believe can reasonably be answered by the user. This task can be thought of as automating the scope and sequencing process used in curriculum and instructional design [14].

The goal of *Question Type Determination* is the specification of a set of characteristics that the question should manifest [5, 10]. This is a somewhat subjective decision, which is based on a dialogue or pedagogical theory and, optimally, on the dialogue context, user model and goals. For example, the pedagogical theory of Questioning the Author [2] prefers types of questions that are open-ended, broad, and require integration of knowledge from across the educational resource. Many educational dialogue strategies start with a deep type of question and, if the student is struggling, gradually move toward shallower levels in Bloom's taxonomy of educational objectives [3]. By contrast, diagnostic QG systems prefer factoid question types in their search for an underlying problem.

Question Realization consists of creating the final natural language question to be posed to the user. The specified question type and target concept identified previously are the primary inputs to this task. Question Realization must either perform syntactic transformations on a source text discussing the target concept or generate a syntactic realization based on a semantic (or similar) representation. Ultimately, it should also consider a user model (including, for example, reading ability), successful and unsuccessful previous interaction styles with this and other users, and the full text or knowledge resource, among other factors.

The focus of this paper is on Target and Key Concept Identification. We discuss issues involved in this task, describe component implementations, present evaluation results demonstrating the task's feasibility, and propose this as a future challenge task for the QG community.

2 Target Concept Identification

Target Concept Identification, identifying the most appropriate concept from which to construct the next question in a dialogue, is arguably the most important task in question generation [9, 18]. If the concept is not important, is the question really worth asking?

While *Target* Concept Identification is performed during the dialogue and is a context-sensitive task, it is also important to identify a priori the set of *key* question-worthy concepts in the knowledge source. Given the application domain, the objective of *Key Concept Identification* is to identify the most important content in the text, that for which questions should be or are likely to be generated. We believe this task is much less sensitive to the application domain

and would, thus, appeal to a broader research community. The more specific focus of this paper is on this Key Concept Identification task and related tasks.

Identifying the key concepts is only one precursor to TCI; in many applications, it is also extremely important to define a logical sequence for introducing these concepts into the dialogue. For example, students would generally find it next to impossible to comprehend a challenging concept without first understanding its more fundamental building-block concepts and hence, it is important for an Intelligent Tutoring Systems (ITS) to consider this in planning its dialogue moves. We refer to this task of defining a progression as Concept Sequencing.

Other subtasks relevant to Key Concept Identification include Concept Relation Identification and Classification, the detection and labeling of inter-concept relationships. For example, knowing that one concept has an *Elaborates* relation with another would be informative to many dialogue strategies.

In the following sections, we discuss system development and evaluation related to these tasks, Key Concept Identification, Concept Sequencing, Concept Relation Identification and Classification.

2.1 Key Concept Identification

Questions can be generated from a variety of knowledge sources, (e.g., plain text, linguistically annotated text, structured databases, and knowledge bases with formal semantic or logic representations such as might be output by a natural language understanding system). The most common of these and the source that our experiments are based on is natural language text, though the majority of our work is equally applicable to other knowledge sources. The goal of Key Concept Identification is to extract important concepts from a knowledge source. In the remainder of this subsection, we detail our center's work in Key Concept Identification and discuss how that ties to the greater goal of target concept identification.

Implementation

The Customized Learning Service for Concept Knowledge (CLICK) is a personalized learning system that assesses a learner's work and recommends digital library resources that can help the learner remedy the diagnosed deficiencies [6]. Key Concept Identification plays a central role in this application's ability to conduct this remediation. The CLICK process of identifying key concepts (Figure 1) consists of three steps: 1) identification of supporting concepts, 2) concept linking and 3) concept graph analysis.

CLICK treats identification of supporting concepts as a multi-document summarization task. De la Chica et al. [4] started with the MEAD [13] summarization system and supplemented it with additional features to ensure fidelity to curricular standards and to select more succinct sentences. These features included a TF-IDF (Term Frequency - Inverse Document Frequency) similarity score between candidate concepts and relevant learning benchmarks. The resulting system, dubbed COGENT, outputs a list of sentences considered to best summarize the salient concepts found within the texts.

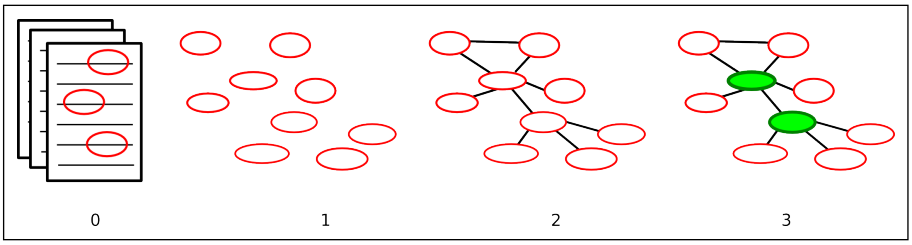


Fig. 1. The steps of CLICK Key Concept Identification: 0-1) extract supporting concepts from digital library resources (web text) using multi-document summarization, 2) identify and classify links between extracted concepts to create a concept map and 3) discover central or key concepts through graph analysis.

To build a concept map, the COGENT output is then fed to an SVM-based link classifier to determine if a candidate pair of concepts should be linked. The link classifier feature set consisted of a variety of lexical, syntactic, semantic, and document structure features. Lastly, Ahmad et al. [1] used the JUNG graph library's[3] implementations of the PageRank and HITS algorithm along with other heuristics to perform a graph analysis to discover the most central or key concepts. The final set of key concepts are the intersection of 1) the set of concepts selected by the PageRank and HITS algorithm, 2) the set of concepts with the highest ratio of incoming versus outgoing links, and 3) the set of concepts with the highest term density.

Experiments

Twenty digital library resources (educational web texts) formed the basis of CLICK key concept identification evaluation. Taken together, experts consider these texts to contain all the information a high school graduate should know about earthquakes and plate tectonics.

To obtain evaluation data, experts were asked to extract and potentially paraphrase spans of text (henceforth called concepts) from each of the 20 resources. Example concepts in the expert map include:

- Concept 19: *Mantle convection is the process that carries heat from the core and up to the crust and drives the plumes of magma that come up to the surface and make islands like Hawaii.*
- Concept 21: *asthenosphere is hot, soft, flowing rock*
- Concept 176: *The Theory of Plate tectonics*
- Concept 224: *a plate is a large, rigid slab of solid rock*
- Concept 610: *P-wave (primary)*
- Concept 608: *S-wave (secondary)*

Four experts were then asked to link the concepts and label the relations (described further in section 2.3), resulting in a map for each resource. The

[3] http://jung.sourceforge.net/

experts then combined their resource maps to span the whole domain. Finally, using the CMAP[4] concept map editing tool, they collaboratively merged their individual maps into a single gold-standard, full-domain concept map.

De la Chica et al. [4] input the 20 digital library resources into COGENT to automatically produce a large collection of domain concepts, which was evaluated against the final expert domain concept map. Ahmad et al. produced a concept map from the COGENT-produced concepts using a concept link classifier. This was evaluated against links in the expert produced concept map. Lastly Ahmad et al. [1] performed graph analysis on the gold-standard expert domain map to identify the core or key concepts versus optional or supporting concepts.

Results and Discussion

To evaluate the COGENT-produced set of concepts, De la Chica et al. [4] used the ROUGE-L metric and cosine similarities to measure the differences between expert-selected and system identified concepts. The COGENT system achieved a ROUGE-L F-Measure of 0.6001 (P=0.5982, R=0.6021) and cosine similarity of 0.8325, while the baseline system, MEAD's default configuration, scored a ROUGE-L F-Measure of 0.5248 (P=0.5623, R=0.4919) and cosine=0.6748. These results do suggest that the summarization approach provides a good first step in generating the full concept map.

Ahmad et al.'s [1] best performing link classifier scored a 0.9651 accuracy on determining whether a link existed between a given pair of concepts, however this score does not tell the full story as the majority of concept pairs should have no link. Thus, the classifier's high accuracy is predominantly a byproduct of this imbalance, and consequently its precision (0.2061) and recall (0.0153) were too poor to reliably support Key Concept Identification. Rather than using unreliable data, the CLICK researchers opted to use the expert map for the final phase in Key Concept Identification.

The CLICK study was primarily a proof of concept to show that student misconceptions could be highlighted, and consequently it did not have an expert-derived (gold-standard) list of key concepts to use in evaluating system performance. To gauge CLICK's ability to discover central concepts within a concept map, Ahmad et al. [1] asked their panel of experts to collaboratively identify regions of concepts on the final domain map and label them with subtopic-areas. Example subtopic-areas included subjects such as *earthquake wave types*, *tsunamis*, and *the theory of continental drift*. Ahmad et al. checked membership of the identified key concepts and found that they covered roughly 80% of the 25 subtopic-areas – indicating that CLICK found important concept nodes across the entire map rather than in just a small subset of subtopics.

The outcomes from the CLICK study suggest that it is possible to extract supporting concepts from resources, and given a reliable set of links, a system can likely distinguish the key concepts from the supporting ones. Clearly the lack of a gold-standard set of key concepts was a weak area in the CLICK evaluation potentially leaving room for improvement in identifying concept relations or links.

[4] http://cmap.ihmc.us/conceptmap.html

2.2 Concept Sequencing

Given the concepts selected in the Key Concept Identification task, the goal of the Concept Sequencing task is to create a directed acyclic graph, which represents the logical order in which concepts should be introduced in a lesson or tutorial dialogue. This is a partial ordering, as there may be several equally relevant next concepts. For example, a teacher might follow a progression such as 1) defining *pitch* as *the perceived fundamental frequency of a sound*, 2) explaining *a shorter string produces a higher pitch*, 3) explaining *a tighter string produces a higher pitch* and 4) closing with a discussion of *the difference in pitch across each of the strings of a violin and cello*. Though 1 should precede and 4 should follow all other concepts, the sequencing of concepts 2 and 3 is not particularly important and could be reversed.

Implementation

To show the viability of automatically producing concept sequences, we built a system based on the idea that concepts that should precede other concepts will exhibit this behavior with some regularity across the corpus of digital library resources. Thus, if concept A usually preceded other concepts in a large set of documents, it should precede them in the output sequence.

Because concepts often do not appear in their entirety in a document, and because some aspects of a concept may show up earlier than others, we treat the sub-problem of finding concept position within a document like an information retrieval task that depends on measures of semantic similarity. To implement this in our proof-of-concept system we used the Lucene[5] information retrieval library with the standard analyzer and indexed the original 20 CLICK resources at the sentence level. Key concepts to be included within the concept sequence were then treated as search queries. Search results with a similarity score below 0.26 were pruned. This threshold was selected by qualitatively evaluating query results to ensure some degree of relevance. The remaining results were used to build a table mapping a concept to the position of its first occurrence in each of the CLICK resources. Because the index was built at the sentence level, this position was simply the sentence number stored in the index.

With all concept positions identified and tabulated, we computed pairwise comparisons of the concepts' sentence numbers. We then took a vote of these comparisons across all the resources, so for example if the sentence number of concept A precedes the sentence number for concept B in a majority of the resources, we consider A to precede B in general. If a concept did not show up in a resource, that resource was withheld in the voting process. Next, we calculated the total number of times a concept preceded any other concept, and sequenced the concepts according to the number of times they preceded other concepts. Concepts with an identical number of predecessors were considered to be at the same level. We could not create an ordering that considered only individual pairwise ordering or that had perfect fidelity with those orderings because it would introduce loops into the overall graph.

[5] http://lucene.apache.org

Experiment

To evaluate the system's output, a gold-standard concept sequence of key concepts was derived from the CLICK data. Because there was no canonical concept sequence available, the gold-standard was assembled from a set of expert provided remediation strategies for individual student essays. Out of 55 previously defined key concepts, 14 that did not occur in any of the remediation strategies were removed, resulting in 41 concepts in our evaluation. We used the frequency of the expert pairwise remediation sequences in these 41 concepts to determine a preliminary concept sequence. We then removed loops by manually judging the relative importance of the component dependencies and removing those deemed to be errant. Since this sequence was built up from expert remediation sequencing, we believe it to be representative of an ordering produced by an expert.

Results and Discussion

The concept sequence was evaluated over F_1-*measure* of just the gold-standard key concepts and the aligned concepts from the system output. Specifically, we calculated an average *instance recall (IR)* over all of the gold-standard key concepts that have predecessors (the non-initial concepts) and an average *instance precision (IP)* over all of the non-initial system-output concepts that were aligned to gold-standard key concepts. Calculations are based on all of the predecessors (direct and indirect) of each concept. *IR* is the fraction of all gold-standard predecessor concepts that were included in the systems output for the aligned concept, and *IP* is the fraction of all system-output predecessor concepts that were included in the gold-standard for the aligned concept.

More formally, let h and l be the total number of non-initial key concepts in the gold standard and the system output respectively, G_i be all of the predecessors of the i^{th} gold-standard concept, and O_j be the set of all predecessors of the system concept aligned to G_i. Now, recall R and precision P can be calculated using the following formulas:

$$R = \frac{1}{h}\sum_{i=1}^{h} IR_i = \frac{1}{h}\sum_{i=1}^{h} \frac{|G_i \cap O_i|}{|G_i|} \tag{1}$$

$$P = \frac{1}{l}\sum_{j=1}^{l} IP_j = \frac{1}{l}\sum_{j=1}^{l} \frac{|O_j \cap G_j|}{|O_j|} \tag{2}$$

The final F_1-*measure* was F_1=*0.526* (*P=0.383*, *R=0.726*). These results, based on a very rudimentary system, provide an interesting baseline on which to judge the feasibility of a concept sequencing task and later the quality of more complete implementations. The large difference between precision and recall stems from the system's tendency towards creating deeper hierarchies. Whereas the gold-standard sequence has multiple initial nodes and several unique branches in the ordering, the system output has only a single initial node. Additionally, the baseline system does not attempt to find orderings between concepts that share the same number of predecessors, consequently all nodes at a given level

have 100% overlap in their predecessor nodes. In other words, the system's all inclusive approach heavily favors recall, but at the expense of being able to identify the differences critical to precision. Future work to address this discrepancy should make better use of the pairwise comparison data to produce less densely packed orderings. Other areas for investigation include improving the detection of the first occurrence of a concept within the documents through more sophisticated semantic similarity measures.

2.3 Concept Relation Identification and Classification

Beyond knowing the sequence of concepts, understanding the relation between them can provide important information for generation of follow-up questions. In the concept relation identification and classification subtask, the goal is to find and label the links between a concept node with the appropriate relationship. Since we already discussed the issues and challenges of concept relation identification in our description of link identification in section 2.1, we will focus here on the concept relation classification half of the subtask.

While there is no CLICK system for concept relation identification (labeling the links), previous studies [16] have shown that it is feasible to train a computational system to label the discourse relationship between concepts given a predefined list of relationships such as those defined by the Penn Discourse Treebank [11] or Rhetorical Structure Theory (RST) [8].

As discussed in section 2.1, Ahmad et al. [1] utilized four experts to link extracted concepts and label their relations. The concept link-relation labels included a combination of discourse-like relations such as *elaborates*, *cause*, *defines* and domain-specific relations like *technique*, *type of*, and *indicates*. Although the vocabulary for the relation labels was unspecified, analysis of the domain map found that the ten most frequent labels accounted for 64% of the links, suggesting that with some expert refinement, a small subset of the CLICK concept map relations [1] and discourse relations [11, 12] could be used for building a gold-standard.

With a training, development, and test set in place, we anticipate that it would be relatively straightforward to build a classifier to apply labels to links between key concepts, using a combination of lexical and syntactic features.

3 Tutoring Dialogue and Question Realization

Currently, a significant amount of effort is required to craft a dialogue system, such as in intelligent tutoring systems (ITS). Much of the effort centers on creation of dialogue content and flow. The majority of dialogue based ITS are built on top of either finite state networks (FSN) or frame and slot architectures. Designing an appropriate and meaningful set of dialogue states is not a trivial task and often requires both domain knowledge as well as system implementation knowledge. Moreover, authoring questions appropriate to these states can be challenging. Though some tools [7, 17] exist to ease the burden, it remains a

very manual, labor-intensive task. It is more desirable to have the ability to seed an ITS with a map of domain specific concepts. This in turn would enable questions to be generated with respect to how a student's knowledge relates to this ontology, rather than to how the student's knowledge maps onto a predefined dialogue state. The subtasks described here collectively provide an important foundation for driving towards this goal of automatic ITS creation. Specifically, key concepts derived from an identification task could be considered anchor nodes in concept-dialogue space and are roughly equivalent to the states in an FSN. Similarly, the concept sequence graph could be regarded as a default dialogue management strategy that ensures concepts are introduced in a logical, meaningful order. Lastly, the concept relation labels when used in conjunction with the key ideas and concept sequences can provide a useful hint for generating an appropriate follow-up question.

To illustrate how the output from these tasks would ultimately result in automatic question generation, consider the scenario where concept 486 *"an earthquake is the sudden slip of part of the Earth's crust..."* precedes concept 561 *".... When the stress in a particular location is great enough ... an earthquake begins"* with a link labeled with a *caused-by* relation. Suppose in the course of a dialogue, an ITS determined that the student had stated a paraphrase of concept *486*. Given the evidence above, an ITS could direct the conversation toward concept *561* by asking questions such as *Now that you have defined what an earthquake is, can you explain what causes them?* Most importantly the ITS has produced a pertinent, meaningful question.

4 Discussion and Conclusion

We have described a set of tasks known as Target Concept Identification that provide a natural progression leading to the final goal of question realization. Because this task produces knowledge critical to generating relevant, context-dependent questions within a dialogue, we believe it is an important challenge for future research.

We are encouraged by both our new baseline in concept sequencing as well as past work in key concept identification, which together show the applicability of Target Concept Identification as a whole. It is our hope that the QG community will continue to consider the task of QG as not just surface form generation, but as the product of several tasks vital to making QG successful at large.

Acknowledgments

This work was supported by grants from the NSF (DRL-0733322, DRL-0733323, DRL-0835393, IIS-0537194) and the IES (R3053070434). Any findings, recommendations, or conclusions are those of the author and do not necessarily represent the views of NSF or IES.

References

1. F. Ahmad, S. de la Chica, K. Butcher, T. Sumner, and J.H. Martin. Towards automatic conceptual personalization tools. In *Proc 7th ACM/IEEE-CS joint conference on Digital Libraries*. ACM, 2007.
2. I. L. Beck, M. G. McKeown, C. Sandora, L. Kucan, and J Worthy. Questioning the author: A year-long classroom implementation to engage students with text. *The Elementary School Journal*, 98:385– 414, 1996.
3. B.S. Bloom. *Taxonomy of Educational Objectives: The Classification of Educational Goals*. Susan Fauer Company, Inc, 1956.
4. S. de la Chica, F. Ahmad, J.H. Martin, and T. Sumner. Pedagogically useful extractive summaries for science education. In *Proc CoLing*, volume 1, pages 177–184. Association for Computational Linguistics, 2008.
5. A Graesser, V Rus, and Z Cai. Question classification schemes. In *Proc WS on the QGSTEC*, 2008.
6. Q. Gu, S. Chica, F. Ahmad, H. Khan, T. Sumner, J.H. Martin, and K. Butcher. Personalizing the selection of digital library resources to support intentional learning. In *Proc Euro Research and Advanced Technology for Digital Libraries*, 2008.
7. P.W. Jordan, B Hall, M Ringenberg, Y Cue, and C Rosé. Tools for authoring a dialogue agent that participates in learning studies. In *Proc AIED*, pages 43–50, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.
8. W.C. Mann and S.A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.
9. RD. Nielsen. Question generation: Proposed challenge tasks and their evaluation. In *Proc WS on the QGSTEC*, 2008.
10. RD Nielsen, J Buckingham, G Knoll, B Marsh, and L. Palen. A taxonomy of questions for question generation. In *Proc WS on the Question Generation Shared Task and Evaluation Challenge.*, 2008.
11. R Prasad, N Dinesh, A Lee, E Miltsakaki, L Robaldo, A Joshi, and B Webber. The penn discourse treebank 2.0. In *Proc LREC*, 2008.
12. R Prasad and Aravind Joshi. A discourse-based approach to generating why-questions from texts. In *Proc WS on the QGSTEC*, 2008.
13. D. Radev, T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Çelebi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel, and Z. Zhang. Mead - a platform for multidocument multilingual text summarization. In *Proc. LREC 2004*, 2004.
14. C.M. Reigeluth. The elaboration theory: Guidance for scope and sequence decisions. In *Instructional-Design Theories and Models: A New Paradigm of Instructional Theory*. Lawrence Erlbaum Assoc, 1999.
15. V. Rus, Z. Cai, and A.C. Graesser. Question generation: An example of a multi-year evaluation campaign. In *Proc WS on the QGSTEC*, 2008.
16. R. Soricut and D. Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proc HLT/NAACL*, pages 228–235, 2003.
17. S. Susarla, A. Adcock, R. Van Eck, K. Moreno, A. C. Graesser, and the Tutoring Research Group. Development and evaluation of a lesson authoring tool for autotutor. In V. Aleven, U. Hoppe, R. Mizoguchi J. Kay, H. Pain, F. Verdejo, and K. Yacef, editors, *Proc. AIED2003*, pages 378–387, 2003.
18. L. Vanderwende. The importance of being important. In *Proc WS on the QGSTEC*, 2008.
19. Howard Wainer. *Computer-Adaptive Testing: A Primer*. 2000.

Overview of The First Question Generation Shared Task Evaluation Challenge

Vasile Rus[1], Brendan Wyse[2], Paul Piwek[2], Mihai Lintean[1], Svetlana Stoyanchev[2] and Cristian Moldovan[2]

[1] Department of Computer Science/Institute for Intelligent Systems, The University of Memphis, Memphis, TN, 38152, USA
{vrus,mclinten,cmoldova}@memphis.edu
[2] Centre for Research in Computing, Open University, UK
bjwyse@gmail.com and {p.piwek, s.stoyanchev}@open.ac.uk

Abstract. The paper describes the First Shared Task Evaluation Challenge on Question Generation that took place in Spring 2010. The campaign included two tasks: Task A – Question Generation from Paragraphs and Task B – Question Generation from Sentences. Motivation, data sets, evaluation criteria, and guidelines for annotators are presented for both tasks.

Keywords: question generation, shared task evaluation campaign.

1 Introduction

Question Generation is an essential component of learning environments, help systems, information seeking systems, multi-modal conversations between virtual agents, and a myriad of other applications (Lauer, Peacock, and Graesser, 1992; Piwek et al, 2007).

Question Generation has been recently defined as the task (Rus & Graesser, 2009) of automatically generating questions from some form of input. The input could vary from information in a database to a deep semantic representation to raw text.

The first Question Generation Shared Task Evaluation Challenge (QG-STEC) follows a long tradition of STECs in Natural Language Processing: see various tracks at the Text REtrieval Conference (TREC; http://trec.nist.gov), e.g. the Question Answering track, the semantic evaluation challenges under the SENSEVAL umbrella (www.senseval.org), or the annual tasks run by the Conference on Natural Language Learning (CoNLL; http://www.cnts.ua.ac.be/conll/). In particular, the idea of a QG-STEC was inspired by the recent activity in the Natural Language Generation (NLG) community to offer shared task evaluation campaigns as a potential avenue to provide a focus for research in NLG and to increase the visibility of NLG in the wider Natural Language Processing (NLP) community (White and Dale, 2008). It should be noted the QG is currently perceived as a discourse processing task rather than a traditional NLG task (Rus & Graesser, 2009).

Two core aspects of a question are the goal of the question and its importance. It is difficult to determine whether a particular question is good without knowing the context in which it is posed; ideally one would like to have information about what counts as important and what the goals are in the current context. This suggests that a STEC on QG should be tied to a particular application, e.g. tutoring systems. However, an application-specific STEC would limit the pool of potential participants to those interested in the target application. Therefore, the challenge was to find a framework in which the goal and importance are intrinsic to the source of questions and less tied to a particular context/application. One possibility was to have the general goal of asking questions about salient items in a source of information, e.g. core ideas in a paragraph of text. Our tasks have been defined with this concept in mind. Adopting the basic principle of application-independence has the advantage of escaping the problem of a limited pool of participants (to those interested in a particular application had that application been chosen as the target for a QG STEC).

Besides the advantage of a larger pool of potential participants, an application-independent QG STEC would provide a more fair ground for comparison as teams already working on a certain application would not be advantaged as would be the case if the application had been the focus of a STEC. It should be noted that the idea of an application-independent STEC is not new. An example of an application-independent STEC would be generic summaries (as opposed to query-specific summaries) in summarization.

Another decision aimed at attracting as many participants as possible and promoting a more fair comparison environment was the input for the QG tasks. A particular semantic representation would have provided an advantage to groups already working with it and at the same time it would have raised the barrier-to-entry for newcomers. Instead, we have adopted a second guiding principle for the first QG-STEC tasks: no representational commitment. That is, we wanted to have as generic an input as possible. The input to both task A and B in the first QG STEC is raw text.

Task A and B described here fall in the Text-to-Question category of QG tasks identified by The First Workshop on Question Generation (www.questiongeneration.org). The first workshop identified four categories of QG tasks (Rus & Graesser, 2009): Text-to-Question, Tutorial Dialogue, Assessment, and Query-to-Question. Using another categoriztion, tasks A and B are part of the Text-to-text Natural Language Generation task categories (Dale & White, 2007).

It is important to say that the two tasks offered in the first QG STEC were selected among 5 candidate tasks by the members of the QG community. A preference poll was conducted and the most preferred tasks, Question Generation from Paragraphs (Task A) and Question Generation from Sentences (Task B), were chosen to be offered in the first QG STEC. The other three candidate tasks were: Ranking Automatically Generated Questions (Michael Heilman and Noah Smith), Concept Identification and Ordering (Rodney Nielsen and Lee Becker), and Question Type Identification (Vasile Rus and Arthur Graesser).

There is overlap between Task A and B in the first QG STEC. This was intentional with the aim of encouraging people preferring one task to participate in the other. The overlap consists of the specific questions in Task A which are more or less similar with the type of questions targeted by Task B.

Overall, we had 1 submission for Task A and 4 submissions for Task B. The submissions are currently evaluated through a peer-review system for Task B. Task A is evaluted by two external judges as there was only one submission and the peer-review mechanism cannot be applied.

2 TASK A: Question Generation from Paragraphs

1.1 Task Definition

The Question Generation from Paragraphs (QGP) task challenged participants to generate a list of 6 questions from a given input paragraph. The six questions should be at three specificity/scope levels: 1 x broad (entire input paragraph), 2 x medium (one or more clauses or sentences), and 3 x specific (phrase or less). The scope is defined by the portion of the paragraph that answers the question. If multiple questions could be generated at one level, only the specified number should be submitted. That is, if the paragraph answers two broad questions then only one should be submitted at that level.

The Question Generation from Paragraphs (QGP) task has been defined such that it is *application-independent*. *Application-independent* means questions will be judged based on content analysis of the input paragraph.

For this task, questions are considered important if they ask about the core idea(s) in the paragraph. Questions are considered interesting if an average person reading the paragraph would consider them so based on a quick analysis of the contents of the paragraph.

Simple, trivial questions such as *What is X?* or generic questions such as *What is the paragraph about?* were avoided. In addition, implied questions (an example is provided later) were not allowed as the emphasis is on questions triggered and answered by the paragraph. Questions should not be compounded as in *What is ... and who ... ?* Questions must be grammatically and semantically correct and related to the topic of the given input paragraph. Question types (*who/what/why/...*) generated for each paragraph should be diverse, if possible. Unique question types are preferred in the set of returned questions.

1.2 Guidelines for Human Judges

We show next an example paragraph together with six interesting, application-independent questions that could be generated. We will use the paragraph and questions to describe the judging criteria.

A set of five scores, one for each criterion (specificity, syntax, semantics, question type correctness, diversity)to each question. Composite scores will also be assigned. For instance, there will be a composite score per question. That is, each question will be assigned a composite score ranging from 1 (first/top ranked, best) to 4 (lowest rank), 1 meaning the question is at the right level of specificity given its rank (e.g. the

broadest question that the whole paragraph answers will get a score of 1 if in the first position) and also it is syntactically and semantically correct as well as unique/diverse from other generated questions in the set.

Table 1. Example of input paragraph (from http://en.wikipedia.org/wiki/Abraham_lincoln).

Input Paragraph
Abraham Lincoln (February 12, 1809 – April 15, 1865), the 16th President of the United States, successfully led his country through its greatest internal crisis, the American Civil War, preserving the Union and ending slavery. As an outspoken opponent of the expansion of slavery in the United States, Lincoln won the Republican Party nomination in 1860 and was elected president later that year. His tenure in office was occupied primarily with the defeat of the secessionist Confederate States of America in the American Civil War. He introduced measures that resulted in the abolition of slavery, issuing his Emancipation Proclamation in 1863 and promoting the passage of the Thirteenth Amendment to the Constitution. As the civil war was drawing to a close, Lincoln became the first American president to be assassinated.

Table 2. Examples of questions and scores for the paragraph in Table 1.

Questions	Scope
Who is Abraham Lincoln?	*General*
What major measures did President Lincoln introduce?	*Medium*
How did President Lincoln die?	*Medium*
When was Abraham Lincoln elected president?	*Specific*
When was President Lincoln assassinated?	*Specific*
What party did Abraham Lincoln belong to?	*Specific*

The specificity scores are assigned primarily based on the answer span in the input paragraph. The broadest question is the one whose answer spans the entire paragraph. The most specific question is the one whose answer is less than a sentence: a clause, phrase, word, or collocation. Scores will be assigned based on the following rubric: 1 – input paragraph, 2 – multiple sentences, 3 – a clause or less, 4 – trivial/generic, implied, no question (empty question), or undecided, e.g. a semantically wrong question may not be understood well enough to judge its scope. As we expect six questions as output, if one level is missed we encourage participants to generate questions of a narrower scope. For instance, if a broad-scope question cannot be

generated then a multiple-sentence or a specific question should be submitted. This assures that each participant submits as many as six questions for each input paragraph.

Best question specificity scores for six questions corresponding to an input paragraph would be 1, 2, 2, 3, 3, 3. The best configuration of scores (1, 2, 2, 3, 3, 3) would only be possible for paragraphs that could trigger the required number of questions at each scope level, which may not always be the case.

While the initial plan was for the judges to look at the question itself and select themselves the portion of the paragraph that may have triggered the question we opted instead, for practical reasons, to allow the judges to see the span of text submitted by participants for each question and decide based on the span the specificity of the question. The advantage of the initial plan is that the judges' selected text span could be automatically compared to participants' for an automated scoring process.

The syntactic correctness will be judged using the following scores: 1 – grammatically correct and idiomatic/natural, 2 – grammatically correct, 3 – some grammar problems, 4 – grammatically unacceptable.

The semantic correctness will be judged using the following scores: 1 – semantically correct and idiomatic/natural, 2 – semantically correct and close to the text or other questions, 3 – some semantic issues, 4 – semantically unacceptable.

Correctness of question type means the specified type by a participant is agreed upon by the judge. This is a binary dimension: 0 – means the judge agrees with the specified answer type, 1 – means the judge disagrees.

Diversity of question types was also be evaluated. At each scope level, ideally, each question will have a different question type. A question type is loosely defined as being formed by the question word (e.g., *wh*-word or auxiliary) and by the head of the immediately following phrase. For instance, in *What U.S. researcher* …. ? the head of the phrase *U.S. researcher that* follows the question word *What* indicates a person which means the question is actually a *Who* question and not a *What* question. Preference will be given to diversity of question words though. Full question types, i.e. including the head of the phrase following the question word, will be considered in special cases when the use of diverse question words is constrained by the input paragraph, that is, when different question words are hard to employ in order to generate different question types. For instance, some paragraphs may facilitate the generation of true *What* questions, i.e. *What* question types, but not *When* questions. For diversity ratings, we will use the following rubric: 1 – diverse in terms of question type and main body, 2 – diverse in terms of main body, 3 – paraphrase of a previous question, and 4 – similar-to-identical to a previous question.

While full diversity would be ideal, it can be quite challenging for some input paragraphs. For pragmatic reasons, we relaxed the diversity criteria. We assigned the highest score of diversity if at least 50% of the question types in the whole 6-question set are different and the distribution of types is balanced, e.g. 2-*Who*, 2-*What*, and 2-*Where* would be scored higher than 4-*Who*, 1-*What*, and 1-*Where*.

Diversity of body will be evaluated also in terms of answer scope by asking judges to highlight in the input paragraph the fragments that constitute the answer to the question according to their opinion.

We also defined composite scores. In general, composite scores are the average of individual composite scores. An individual composite score summarizes the scores

along a dimension, e.g. syntactic correctness, and is computed by taking the average of individual scores shown by the formula below where Q is the number of questions.

$$Syntactic-overall-score = \frac{\sum_{|Q|} individual\_score}{|Q|}$$

The composite score for specificity is more challenging to define. The goal would be to have a summative score with values from 1 to 4. 1 should be assigned to perfect system that generates 6 questions for each paragraph with the required distribution of specificity levels: 1 general, 2 medium, and 3 specific. For instance, if there are 4 specific questions in a set of 6 questions, then when judging the fourth specific question it will be penalized because the number of expected specific questions (3) have been exhausted and another question at general or medium scope has not been generated. As of this writing, we are refining our composite score for specificity.

1.3 Data Sources and Annotation

The primary source of input paragraphs were: Wikipedia, OpenLearn, Yahoo!Answers. We collected 20 paragraphs from each of these three sources. We collected both a development data set (65 paragraphs) and a test data set (60 paragraphs). For the development data set we manually generated and scored 6 questions per paragraph for a total of 6 x 65 = 390 questions.

Paragraphs were selected such that they are self-contained (no need for previous context to be interpreted, e.g. will have no unresolved pronouns) and contain around 5-7 sentences for a total of 100-200 tokens (excluding punctuation). In addition, we aimed for a diversity of topics of general interest.

We decided to provide minimal annotation for input in order to allow individual participants to choose their own preprocessing tools. We did not offer annotations for lemmas, POS tags, syntactic information, or PropBank-style predicate-argument structures. This linguistic information can be obtained with acceptable levels of accuracy from open-source tools. Furthermore, this favors comparison of full systems in a black-box manner as opposed to more specific components. We only provided discourse relations based on HILDA, a freely available automatic discourse parser (duVerle & Prendinger, 2009).

2 TASK B: Question Generation from Sentences

2.1 Task Definition

Participants were given a set of inputs, with each input consisting of:

- a single sentence and

- a specific target question type (e.g., WHO?, WHY?, HOW?, WHEN?; see below for the complete list of types used in the challenge).

For each input, the task was to generate 2 questions of the specified target question type.

Input sentences, 60 in total, were selected from OpenLearn, Wikipedia and Yahoo! Answers (20 inputs from each source). Extremely short or long sentences were not included. Prior to receiving the actual test data, participants were provided with a development data set consisting of sentences from the aforementioned sources and, for one or more target question types, examples of questions. These questions were manually authored and cross-checked by the team organizing Task B.

The following three examples are taken from the development data set, one example from OpenLearn, Wikipedia and Yahoo! Answers. Each instance has a unique identifier and information on the source it was extracted from. The <text> element contains the input sentence and the <question> elements contain possible questions. The <question> element has the type attribute for specification of the target question type.

```
<instance id="3">
        <id>OpenLearn</id>
        <source>A103_5</source>
        <text>
                The poet Rudyard Kipling lost his only son
                in the trenches in 1915.
        </text>
        <question type="who">
                Who lost his only son in the trenches in 1915?
        </question>
        <question type="when">
                When did Rudyard Kipling lose his son?
        </question>
        <question type="how many">
                How many sons did Rudyard Kipling have?
        </question>
</instance>

<instance id="46">
        <id>Wikipedia</id>
        <source>Igneous_rock</source>
        <text>
                Two important variables used for the classification of
                igneous rocks are particle size, which largely depends
                upon the cooling history, and the mineral composition
                of the rock.
        </text>
        <question type="which">
                Which two important variables are used for the
```

```
            classification of igneous rocks?
        </question>

    <instance id="77">
        <id>YahooAnswers</id>
        <source>
            http://answers.yahoo.com/question/index;_ylt=AoWV
            WMdwLigujQeRxf0LHWCD6xR.;_ylv=3?qid=201002200
            15521AA0slZo
        </source>
        <text>
            In Australia you no longer can buy the ordinary
            incandescent globes, as you probably already know.
        </text>
        <question type="where">
            Where can you no longer buy the ordinary incandescent globes?
        </question>
        <question type="yes/no">
            Can you buy the ordinary incandescent globes in Australia?
        </question>
        <question type="what">
            What can you no longer buy in Australia?
        </question>
        </instance>
```

Note that input sentences were provided as raw text. Annotations were not provided. There are a variety of NLP open-source tools available to potential participants and the choice of tools and how these tools are used was considered a fundamental part of the challenge.

Participants were also provided with the following list specifying the target question types:

- WHO?: The answer to the generated question is a person (e.g. Abraham Lincoln) or group of people (e.g. the American people) named in the input sentence.
- WHERE?: The answer to the generated question is a placename (e.g. Dublin, Mars) or location (North-West, to the left of) which is contained in or can be derived from the input sentence.
- WHEN?: The answer is a specific date (e.g. 3rd July 1973, 4th July), time (e.g. 2:35, 10 seconds ago), era or other representation of time.
- WHICH?: The answer will be a member of a category (e.g. Invertebrate or Vertebrate) or group (e.g. Colours, Race) or a choice of entities (e.g. Union or Confederacy) given in the input sentence.
- WHAT?: The question might describe a specific entity mentioned in the input sentence and ask what it is. The question may also ask the purpose, attributes or relations of an entity as described in the input sentence.

- WHY?: The question asks the reasoning behind some statement made in the input sentence
- HOW MANY/LONG?: The answer will be a duration of time or range of values (e.g. 2 days) or a specific count of entities (e.g. 32 counties) within the input sentence.
- YES/NO: The generated question should ask whether a fact contained in the input sentence is either true or false (e.g. Are mathematical co-ordinate grids used in graphs?).

2.2 Evaluation criteria for System Outputs and Human Judges

The evaluation criteria fulfilled two roles. Firstly, they were provided to the participants as a specification of the kind of questions that their systems should aim to generate. Secondly, they also played the role of guidelines for the judges of system outputs in the evaluation exercise.

For this task, five criteria were identified:

- Relevance
- Question Type
- Syntactic Correctness and Fluency
- Ambiguity
- Variety

All criteria are associated with a scale from 1 to *N* (where *N* is 2, 3 or 4), with 1 being the best score and *N* the worst score.

The criteria are defined as follows:

Relevance

Questions should be relevant to the input sentence. This criterion measures how well the question can be answered based on what the input sentence says.

Table 3. Scoring rubric for relevance.

Rank	***Description***
1	The question is completely relevant to the input sentence.
2	The question relates mostly to the input sentence.
3	The question is only slightly related to the input sentence.
4	The question is totally unrelated to the input sentence.

Question Type

Questions should be of the specified target question type.

Table 4. Scoring rubric for Question Type.

Rank	*Description*
1	The question is of the target question type.
2	The type of the generated question and the target question type are different.

Syntactic Correctness and Fluency

The syntactic correctness is rated to ensure systems can generate grammatical output. In addition, those questions which read fluently are ranked higher.

Table 5. Scoring rubric for syntactic Correctness and Fluency.

Rank	*Description*	*Example*
1	The question is grammatically correct and idiomatic/natural.	In which type of animals are phagocytes highly developed?
2	The question is grammatically correct but does not read as fluently as we would like.	In which type of animals are phagocytes, which are important throughout the animal kingdom, highly developed?
3	There are some grammatical errors in the question.	In which type of animals **is** phagocytes, which are important throughout the animal kingdom, highly developed?
4	The question is grammatically unacceptable.	**On** which type of animals **is** phagocytes, which are important throughout the animal kingdom, developed?

Ambiguity

The question should make sense when asked more or less out of the blue. Typically, an unambiguous question will have one very clear answer.

Table 6. Scoring rubric for Ambiguity.

Rank	*Description*	*Example*
1	The question is unambiguous.	Who was nominated in 1997 to the U.S. Court of Appeals for the Second Circuit?
2	The question could provide more information.	Who was nominated in 1997?
3	The question is clearly ambiguous when asked out of the blue.	Who was nominated?

Variety

Pairs of questions in answer to a single input (i.e., with the same target question type) are evaluated on how different they are from each other. This rewards those systems which are capable of generating a range of different questions for the same input.

Table 7. Scoring rubric for Variety.

Rank	*Description*	*Example*
1	The two questions are different in content.	Where was X born?, Where did X work?
2	Both ask the same question, but there are grammatical and/or lexical differences.	What is X for?, What purpose does X serve?
3	The two questions are identical.	

The procedure for applying these criteria is as follows:

- Each of the criteria is applied *independently* of the other criteria to each of the generated questions (except for the stipulation provided below).

We need some specific stipulations for cases where no question is returned in response to an input. For each target question type, two questions are expected. Consequently, we have the following two possibilities regarding missing questions:

- *No question is returned for a particular target question type*: for each of the missing questions, the worst score is recorded for all criteria.
- *Only one question is returned*:[1] For the missing question, the worst score is assigned on all criteria. The question that is present is scored following

[1] This includes cases where exactly the same question is returned twice for a target question type. In that case, the second identical question is treated the same way as a missing question.

the criteria, with the exception of the VARIETY criterion for which the lowest possible score is assigned.

The result of applying these criteria to a set of questions *including* the missing questions $q_1,\ldots, q_n$ is a score for each (missing) question q_k ($1 \leq k \leq n$) for each of the criteria:

- ScoreRelevance(q_k)
- ScoreQuestionType(q_k)
- ScoreCorrectness(q_k)
- ScoreAmbiguity(q_k)
- ScoreVariety(q_k)

We compute the overall score on a specific criterion C given $q_1,\ldots, q_n$ as follows:

$$\text{OverallScoreC}(q_1,\ldots, q_n) = \sum_{k=1}^{n} \text{ScoreC}(q_k)$$

This way we can, for example, rank systems on the Relevance criterion: the system with the lowest OverallScoreRelevance is ranked first.

We can also compute a score which aggregates the overall scores for the criteria. Here we, however, need to be careful since some criteria are more important than others. For example, if a generated question is irrelevant or of the wrong question type, the scores for correctness, ambiguity and variety don't really matter. Otherwise, a system could achieve high scores through the trivial strategy of always generating the *same* pair of correct/fluent, unambiguous and different questions.

Thus, we propose to calculate the aggregate score as follows:

- For each individual question, if the score for Relevance is 4 or the score for Question Type is 2, revise the score for all the other criteria to the worst possible score.
- Now, compute the OverallScoreC for each criterion C and add these together to obtain the aggregate score.

Of course, we acknowledge that the computation of the aggregate score does have an element of arbitrariness. For instance, we could, alternatively, have assigned weights to the different criteria and used those in the calculation of the aggregate score (then again, the precise choice of the weights would be a further contentious issue). To take this into account, when reporting the Task B results, we will not just return a single total score for each system, but rather provide a profile for each of the evaluated systems and systematic comparisons between them. This way we hope to provide a better insight into which aspects of QG each of the systems is good at.

Conclusions

The submissions to the first QG STEC are being evaluated as of this writing using peer-review mechanism in which participants blindly evaluate their peers questions.

At least two reviews per submissions are performed with the results to be made public at the 3rd Workshop on Question Generation that will take place in June 2010.

Acknowledgments. We are grateful to a number of people who contributed to the success of the First Shared Task Evaluation Challenge on Question Generation: Rodney Nielsen, Amanda Stent, Arthur Graesser, Jose Otero, and James Lester. Also, we would like to thank the National Science Foundation who partially supported this work through grants RI-0836259 and RI-0938239 (awarded to Vasile Rus) and the Engineering and Physical Sciences Research Council who partially supported the effort on Task B through grant EP/G020981/1 (awarded to Paul Piwek). The views expressed in this paper are solely the authors'.

References

1. Lauer, T., Peacock, E., & Graesser, A. C. (1992) (Eds.). *Questions and information systems*. Hillsdale, NJ: Erlbaum.
2. Rus, V. and Graesser, A.C. (2009). *Workshop Report: The Question Generation Task and Evaluation Challenge*, Institute for Intelligent Systems, Memphis, TN, ISBN: 978-0-615-27428-7.
3. Piwek, P., H. Hernault, H. Prendinger, M. Ishizuka (2007). T2D: Generating Dialogues between Virtual Agents Automatically from Text. In: Intelligent Virtual Agents: Proceedings of IVA07, LNAI 4722, September 17-19, 2007, Paris, France, (Springer-Verlag, Berlin Heidelberg) pp.161-174
4. Dale, R. & M. White (2007) (Eds.). *Position Papers of the Workshop on Shared Tasks and Comparative Evaluation in Natural Language Generation*.
5. duVerle, D. and Prendinger, H. (2009). A novel discourse parser based on Support Vector Machines. Proc 47th Annual Meeting of the Association for Computational Linguistics and the 4th Int'l Joint Conf on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP'09), Singapore, Aug 2009 (ACL and AFNLP), pp 665-673.

Automation of Question Generation From Sentences

Husam Ali, Yllias Chali, and Sadid A. Hasan

University of Lethbridge, Lethbridge, AB, Canada
{husam.ali,yllias.chali,sadid.hasan }@uleth.ca

Abstract. Question Generation (QG) and Question Answering (QA) are key challenges facing systems that interact with natural languages. The potential benefits of using automated systems to generate questions helps reduce the dependency on humans to generate questions and other needs associated with systems interacting with natural languages. In this paper we consider a system that automates generation of questions from a sentence, given a sentence, the system, will generate all possible questions which this sentence contain these questions answers. Since the given sentence may be a complex sentence, the system will generate elementary sentences, from the input complex sentences, using a syntactic parser. A part of speech tagger and a named entity recogniser are used to encode needed information. Based on the subject, verb, object and preposition the sentence will be classified, in order determine the type of questions that can possibly be generated from this sentence. We use development data provided by the Question Generation Shared Task Evaluation Challenge 2010.

Keywords: Question Generation, Syntactic Parsing, POS Tagging, Elementary Sentence, Named Entity Tagging, Recall.

1 Introduction

Learners, who actively self-regulate their learning, are often question generators [5]. Given that they recognize their knowledge deficits, learners ask questions that are either triggered by their deficits, or seeking reliable information sources to answer their questions. Unfortunately, this idealistic vision of intelligent inquiry is rarely met, except for the most skilled learners, as most learners have trouble identifying their own knowledge deficits [5].

In this paper we considered a Text-to-Question generation task using syntactic parsing, Part Of Speech (POS) tagger and Named Entity analyzer.

We had different modules to process the raw data, to generate elementary sentences from complex sentences using syntactical parser, Part of Speech tagger and Named Entity analyzer.

The elementary sentences got syntactically parsed in the next module, and using the Part of Speech tagged and Named Entity analyzed representation of the elementary sentences, we classified the sentences to determine the possible questions types that can be generated.

The questions generated got ranked based on the structure of the elementary sentence.

In the subsequent sections of this paper we will discuss the related work (section 2), the system and its structure (section 3), the evaluation of the system results (section 4) and conclusions and future plans (section 5).

2 Related Work

In the field of computational linguistics, dealing with Question Generation (QG) is getting more attention from the researchers [6]. Before the internet and electronic data storage, the time to search and find an answer for questions could extend for weeks hunting for documents in the library. Electronic books and information sources will be the mainstream in the future. In the last few years, new preoccupations appeared for automatic question generation. In ICITA'05 [1], they introduced a template-based approach to generate questions on four types of entities. The authors in ASEE/IEEE Frontiers in Education Conference [3] used WTML (Web Testing Markup Language), which is an extension of HTML, to solve the problem of presenting students with dynamically generated browser-based exams with significant engineering mathematics content.

3 Sentence to Question Generation

This section will discuss the overall framework for the Question Generation (QG) system. We considered the Question Generation from a single input sentence with a question type target (e.g. Who? Where? When? Etc,). For this purpose we used the development data provided by Question Generation Shared Task Evaluation Challenge 2010 (QGSTEC). Sentences will be selected from primary data sources for the QGSTEC (Wikipedia, Yahoo! Answers and OpenLearn) where extremely long or short sentences will be avoided. Since the sentences might have a complex structure with multiple clauses, it would be difficult to generate accurate questions from the complex sentences. Therefore, using syntactic information we simplify the process by extracting elementary sentences from the complex sentences. Based on the sentences subject, verb, object and preposition we classify the sentences to determine the possible type of questions that will be generated. Figure 1 depicts the basic structure of our QG system.

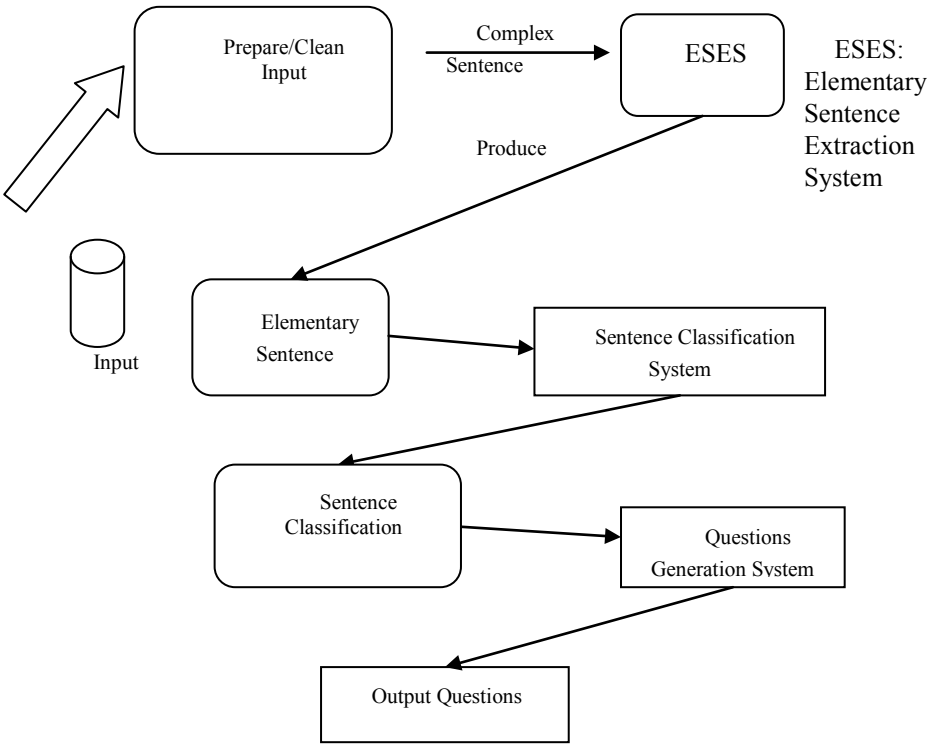


Fig. 1. Overview diagram for the QG system

We divide the work tasks into four modules:

3.1 Data Preparation

An important initial part of any NLP task is cleaning and processing the raw data. We remove the redundant tags and text from the document that has the input sentences that is provided by (QGSTEC). To tokenize the sentences we use the Oak system[1]. The system opens the file that contains the sentences, and reads the sentences. Then, we pass the sentences to the Part of Speech (POS) tagger and the Named Entity (NE) tagger. To generate the POS tagged sentence we use the Oak system. We get the information about the verbs and their tenses from the POS tagged sentences. We again use the Oak system to generate the NE tagged sentences. The Oak system provides us with 150 NEs possible types such as: PERSON, LOCATION, ORGANIZATION, DATE, TIME, MONEY, COUNT, etc. Oak system does not just recognize and label proper nouns, it also is able to recognize common nouns such as

[1] http://nlp.cs.nyu.edu/oak/

school, ship and drug. Oak also can label offences such as first-degree murder and position titles such as King, CEO and president

3.2 Elementary Sentence Construction

The provided sentences by (QGSTEC), may include complex grammatical structure with embedded clauses. Because of this to generate more accurate questions, we extracted the elementary sentences from the complex sentences. To attain this we syntactically parse each complex sentence. Syntactic parsing is analyzing a sentence using the grammar rules. We use Charniak parser[2]. This module constructs a syntactic tree representation, from the bracketed representation of the parsed sentence. While building the tree process, we construct 3 arrays, one for the Noun Phrases (NPs), one for the Verb Phrases (VPs) and one for the Prepositions (PPs) with their location in the tree, a fourth array is generated, from the tree to represent the depth first sequence of the tree nodes and leaves structure. We combine the NPs with the VPs and PPs by reading the NPs till the scope of the VPs and the PPs that are in the VPs scope and thus, we get the elementary sentences. The depth of the first sequence helps us to determine if the phrases to be joined are sequentially correct with the respect of the sentence structure. As an example, "Tom eats an apple and plants a tree". The depth of the first sequence check will prevent the system from generating the elementary sentence, "Tom plant an apple" since the verb plant came after the noun apple.

3.3 Sentence Classification

In this module the input is the elementary sentences. Using the syntactic parser to parse the elementary sentence, and based on the associated POS and NE tagged information, we get from each elementary sentence the subject, object, preposition and verb. This information is used to classify the sentences. This module has two simple classifiers. The first classifies the sentence into fine classes (Fine Classifier) and the second classifies the sentences into coarse classes (Coarse Classifier). This approach is similar to the one that was described in the Journal of Natural Language Engineering [2] but opposite to the approach that was used by Li & Roth (2006). The second classifier will use the first classifier where candidate labels are generated by reducing the set of taken fine classes from the first into a set of coarse classes. This set is treated as the confusion set for the first classifier. OAK system has 150 named entity types that can be tagged. They are included in a hierarchy. This information is used to make candidate fine and coarse classes. We define the five major coarse classifications as:

1. Human: This will have any subject that is the name of a person.
2. Entity: This includes animals, plant, mountains and any object.

[2] Available at ftp://ftp.cs.brown.edu/pub/nlparser/

3. Location: This will be the words that represent locations, such as country, city, school, etc.
4. Time: This will be any time, date or period such as year, Monday, 9 am, last week, etc.
5. Count: This class will hold all the counted elements, such as 9 men, 7 workers, measurements like weight and size, etc.

Organizations which include companies, institutes, government, market, etc are all a type of category Entity in our classification. Once the sentence words have been classified to coarse classes, we consider the relationship between the words in the sentence. As an example, if the sentence has the structure "Human Verb Human", it will be classified as "whom and who" question types. If it is followed by a preposition that represents date, then we add the "When" question type to its classification.

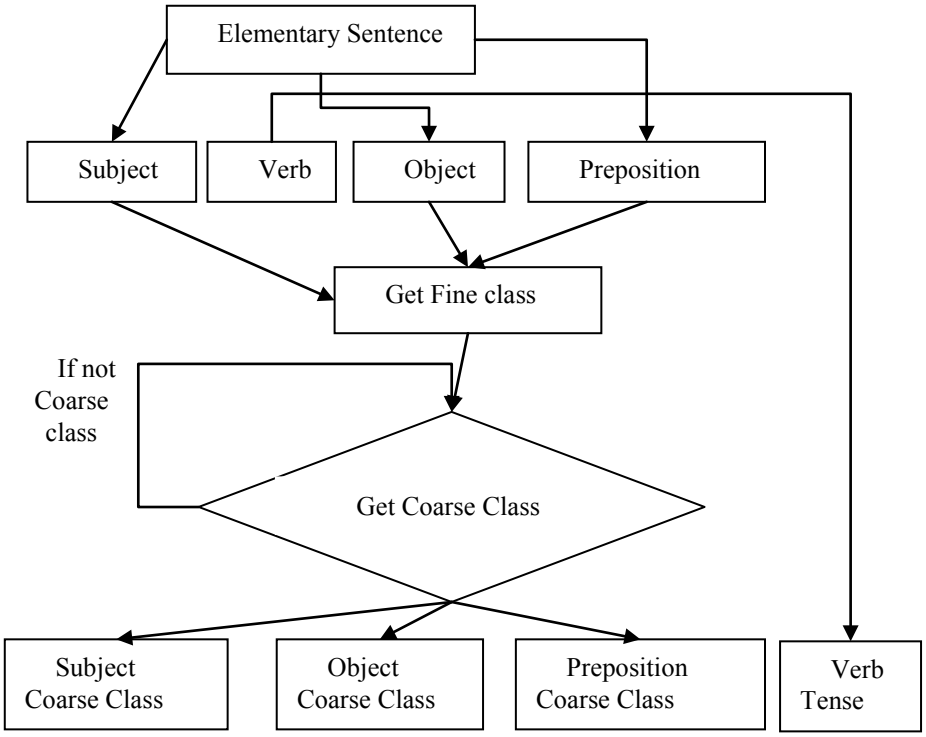


Fig. 2. Coarse Classes and Fine Classes classification diagram

In Fig. 3 we show a sample of the process of classifying a Named Entity type into coarse class Entity.

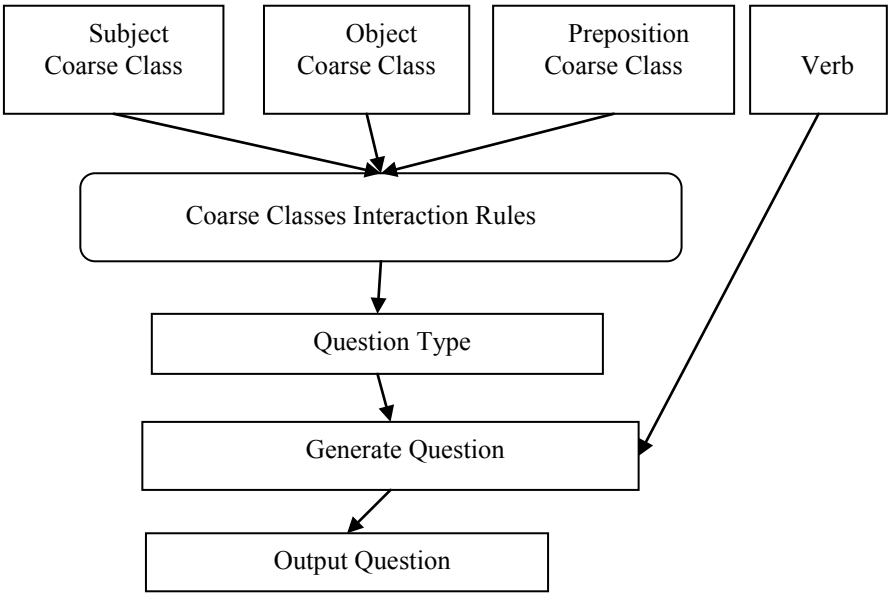


Fig. 3. Question Generation process diagram

3.4 Question Generation

This module takes the elements of the sentences with their coarse classes, the verbs (with its stem) and the tense information. Based on a set of 90 predefined interaction rules, we check the coarse classes according to the word to word interaction.
For example:

Tom ate an orange at 7 pm

Tom is a subject of coarse class Human
An orange is an object of type Entity
At 7 pm is a preposition of type Time

Sample generated questions based on the rule “Human Entity Time” will be:

Who ate an orange?
Who ate an orange at 7 pm?
What did Tom eat?
When did Tom eat an orange?

A sample of an interaction rules is shown in Table 1

Core classes: H = Human E= Entity L= Location T=Time C=Count

Relations			Question type	Example Questions
Subject	Object	Preposition		
H	H	-	Who	Who teach Tom?
			Whom	Whom Sam teaching?
			What	What did Sam do to Tom?
H	H	L	Who	Who teach Tom?
			Whom	Whom Sam teaching?
			What	What did Sam do to Tom?
			Where	Where did Sam teach Tom?
H	L	T	Who	Who study at U of L?
L	H		Where	Where does Sam study?
			When	When did Sam study at U of L?
C	C	-	How many	How many farmers plant 10 trees?
			How many	How many trees did the 10 farmers plant?
E	E	L	Who	Who bought IBM?
			What	What the rabbit eat?
			Where	Where did the rabbit eat the carrot?

Table 1. Sample interaction rules

4 Evaluation Results

For the evaluation of the system, we ran it against the development data provided by (QGSTEC). The development data were provided in the format of a single sentence and a specific target question type (E.g. WHO? WHY? HOW? WHEN? etc). Sentences were selected from the primary data sources for the QGSTEC (Wikipedia, Yahoo! Answers and OpenLearn). In warrants mentioning that extremely long or short sentences were avoided. We used the questions provided by (QGSTEC) for each sentence from different types of possible questions for the sentence. We then employed the widely used evaluation measure: Recall. We define Recall as follows:

$$Recall = \frac{Qg \cap Qa}{Qa}$$

Where, Qg is the number of question generated by our QG system and Qa is the number of actual questions provided by (QGSTEC). With consideration for the fact that humans sometimes generate questions that are worded differently than the sentence structure, results can be seen in Table 2 which represents the recall for the

different types of questions that the system was able to generate. We also included the overall recall for the system results:

Type	Qg	Qa	Recall
What	14	36	0.389%
Which	3	30	0.100%
Who	15	25	0.600%
Where	7	23	0.304%
When	11	29	0.379%
How	3	21	0.143%
Why	2	8	0.250%
Yes/No	2	9	0.222%
Over all Recall	57	181	0.315%

Table 2. Evaluation Results

We found that type "Who" had the highest recall, while types "What, Who, Where and When" were in a closed range between above 0.300.

However the other types were not as good, since they had a recall below 0.300%.

The overall recall for the results was 0.315%, which can be improved if we included the semantically parsed sentences, in the process of generating the elementary sentences, and the generation of the questions to adopt the possibilities of wording the questions differently while preserving the meaning.

The noise that was generated was high considering the provided sample questions. The noise was due to both grammatical incorrectness and questions that were generated, which are not included in the dataset provided, but the grammatically correct generated questions were high.

We also used the other widely used method of evaluation which is precision. We calculated the precision for the factoid questions types, and we found that it is better to use the other widely used method of evaluation measure: Precision. We define Precision as follow:

$$Precision = \frac{Qg \cap Qr}{Qr}$$

We conducted experiment with 20 sentences for the precision evaluation due to the human effort needed for this kind of evaluation method.

Question Type	Qr	Qg ∩ Qr	Precision
What	49	19	0.388
Which	31	7	0.226
Who	70	45	0.643
Where	55	31	0.564
When	53	27	0.509
How many/ How Much	43	17	0.395

Table 3. Evaluation results for the different types of factoid questions that were generated by our system using Precision

Question Type	Qg	Qg ∩ Qr	Precision
Overall Factoid	301	146	0.485

Table 4. Overall evaluation results for the different types of factoid questions that were generated by our system using Precision

5 Conclusion and Future Work

In this paper, we proposed an approch to automatically generate questions given a sentence. We used the development data given by (QGSTEC) and evaluated our system using Recall. We used human effort to evaluate the system. We extracted elementary sentences from complex sentences using syntactic information and classified the elementary sentences. We generated questions based on the subject, verb, object and prepostion using a predefined interaction rules. We plan to extend the number of interaction rules. We will also focus on the sentence classification module to make it more robust. Since human generated questions tend to have words with different meanings and senses, the system can be improved with the inclusion of semantic information and word sense disambiguation. We hope to develop further mechanisms to QG based on the dependency features of the answer and its finding [2][4].

6 References

1. Andrenucci, A. & Sniders, E. (2005). Automated Question Answering: Review of the Main Approaches. In Proceedings of the 3rd International Conference on Information Technology and Applications (ICITA'05), Sydney, Australia.
2. Li, X. & Roth, D. (2006). Learning Question Classifiers: The Role of Semantic Information. Jurnal of Natural Language Engineering, 12(3), 229-249
3. Mcgough, J., Mortensen, J., Johnson, J. & Fadali, S. (2001). A Web-based Testing System with Dynmic Question Generation. In ASEE/IEEE Frontiers in Education Conference.

4. Pinchak, C. & Lin, D. (2006). A Probabilistic Answer Type Model. In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, p. 393-400.
5. Rus, V. & Grasser, A. C. (2009). The Question Generation Shared Task and Evaluation Challenge. In Workshop on the Question Generation Shared Task and Evaluation Challenge, Final Report, The University of Memphis: National Science Foundation, http://www.cs.memphis.edu/~vrus/questiongeneration/TheQuestionGenerationSharedTaskAndEvaluationChallenge.pdf.
6. Leung, H., Li, F. & Lau, R. Advances in Web Based Learning - ICWL 2007: 6th International Conference

Question Generation with Minimal Recursion Semantics

Xuchen Yao† and Yi Zhang†‡

†Department of Computational Linguistics & Phonetics, Saarland University
‡LT-Lab, German Research Center for Artificial Intelligence
Saarbrücken, Germany
{xuchen,yzhang}@coli.uni-saarland.de

Abstract. This paper proposes a semantics-based approach to question generation by transforming the Minimal Recursion Semantics representation of declarative sentences to that of interrogative sentences. Sentences are first decomposed into smaller units on the semantic level. Then the semantic representation of target words are mapped to that of question words. Finally generation is realized through a linguistically deep English grammar. A prototype system is developed to verify the feasibility.

Key words: question generation, deep linguistic grammar, Minimal Recursion Semantics.

1 Introduction

Question Generation (QG) is the task of generating reasonable questions from a text. In terms of target complexity, the types of QG can be divided into *deep* QG (with deep questions, such as *why, what-if, how* questions) and *shallow* QG (with shallow questions, such as *who, what, when, where, which, how many/much, yes/no* questions) ([12]).

Different systems have been proposed or implemented to facilitate QG research and applications. These systems can be divided into mainly three categories: template-based ([9]), syntax-based ([14], [8]) and semantics-based ([13]). Template-based approaches are mostly suitable for applications with a special purpose, which sometimes comes within a closed-domain. The tradeoff between coverage and cost is hard to balance because human labors must be involved to produce high-quality templates. Syntax-based approaches are rather effective, especially for short sentences. The whole generation is based on tree nodes matching and operation. All operations are straight-forward from a syntactic point of view. However, the computer does this without knowing any underlying meaning of the transformed sentence. Also, it sometimes generates ungrammatical questions, which come from the surface realization of transformed trees thus do not always guarantee grammaticality.

While the first two kinds have already been applied, the third approach is theoretically more interesting and practically challenging. Following [13], we propose a semantics-based method of transforming the Minimal Recursion Semantics (MRS, [4]) representation of declarative sentences to that of interrogative

sentences. As a subtask of semantics-based QG, the main efforts would be put to develop algorithms of decomposing complex semantic representations into smaller units and relations between them, which can be reused by other NLP tasks such as natural language understanding.

A set of tools from the DELPH-IN[1] community are assembled to help fulfill this purpose. Specifically, the MRS analysis is obtained from PET (a platform for experimentation with efficient HPSG processing techniques, [1]) while the generation function goes to the Linguistic Knowledge Builder (LKB, [5]). The underlying core linguistic component is the English Resource Grammar (ERG, [7]), a precision-oriented broad-coverage linguistic grammar in the framework of HPSG ([11]). Section 2 gives a more general introduction of these components while Section 3 specifies the system architecture. We address future work and conclude in Section 4.

2 Background

Minimal Recursion Semantics (MRS, [4]) is a meta-level language for describing semantic structures in some underlying object language. It can be implemented in typed feature structures and a bag of Elementary Predications (EPS) are its main components. An Elementary Predication (EP) is a single relation with its arguments, such as `chase(e,x,y)`, while the first argument (or ARG0), `e`, is called the bound variable. In the context of specific grammars that has an MRS representation, such as the ERG, the bound variable `e` in `chase(e,x,y)` denotes an event, following a form of Davidsonian representation in which all verbs introduce events. The capability of underspecifying out-scoping relations is an attractive feature of MRS and makes it a convenient semantic representation in large scale grammar engineering.

The English Resource Grammar (ERG, [7]) is a general-purpose broad-coverage grammar implementation under the HPSG framework. It consists of a large set of lexical entries under a hierarchy of lexical types, with a modest set of lexical rules for production. The Linguistic Knowledge Builder (LKB, [5]) is a grammar development environment for grammars in typed feature structures and unification-based formalisms. It can examine the competence and performance of a grammar by the means of parsing and generation. Central in our task, once given a valid MRS representation, linguistic realizations can be accomplished by chart generation ([3], [2]) in LKB.

Although ERG has a wide coverage of lexicons, there are always unknown words in real texts. Thus a high performance parsing system which incorporates statistical robust processing techniques is needed. We use PET ([1]), a platform for experimentation with efficient processing of unification-based grammars, to handle the parsing task efficiently and robustly. It employs a two-stage parsing model ([15]) with HPSG rules and PCFG models, balancing between precise linguistic interpretation and robust probabilistic coverage.

[1] Deep Linguistic Processing with HPSG: `http://www.delph-in.net/`

3 System Architecture

This section mainly describes the pipelines of the semantics-based question generator. It elaborates the core components: MRS decomposition for complex sentences and MRS transformation for simple sentences by examples. At last language independence and domain adaptability are addressed.

3.1 Overview

A prototype system based on semantics, MrsQG[2], is developed to constructs a general framework and test field for question generation on MRS, including modularized pre-processing, MRS manipulation, parsing and generation etc. Fig. 1 shows the processing pipeline. The following is a brief description of each step.

1. Term extraction. Terms are extracted as answer candidates. The Stanford Named Entity Recognizer ([6]), a regular expression NE tagger, an Ontology NE tagger are used to extract terms.
2. FSC construction. The Feature Structure Chart (FSC) format[3] is an XML-based format that introduces tokenization and external annotation to the ERG grammar and PET parser. Using FSC makes the terms annotated by NERs known to the parser. Thus all terms, no matter how long it is, are treated as an un-splittable token in the initial parsing chart.
3. Parsing with PET. PET accepts FSC and outputs MRS structures. Individual components are communicated through internal XML representations.
4. MRS decomposition. For complex sentences, it needs to be first broken into shorter ones with valid and meaningful semantic representation. Also, this shorter semantic representation must be able to generate outputs. Details in Section 3.2.
5. MRS transformation. With a valid MRS of a sentence, transformations are made to replace EPS for named entities with EPS for (WH) question words. Details in Section 3.3.
6. Generating with LKB. A two-phase generation algorithm ([3], [2]) is used.
7. Output selection. From a well-formed MRS, LKB might give multiple output. Some output might not sound fluent due to the fact that the ERG generates all linguistically plausible realizations. Thus various ranking guidelines are implemented to select the best one.
8. Output to console/XML. Depending on the purpose, MrsQG outputs to console for user interaction or XML files for formal evaluation.

The following sections describes more on step 4 and 5.

[2] http://code.google.com/p/mrsqg/
[3] http://wiki.delph-in.net/moin/PetInputFsc

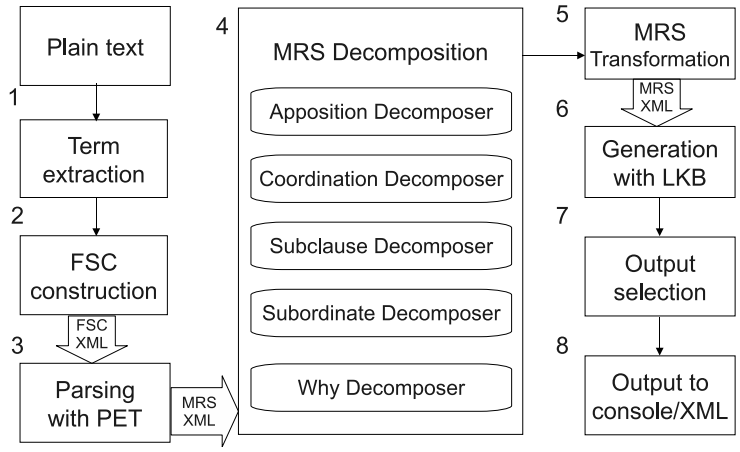
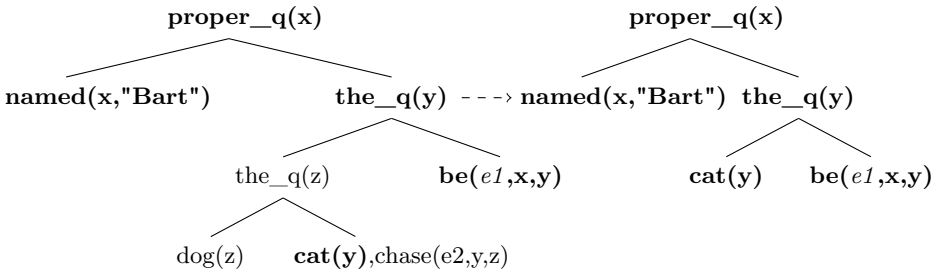
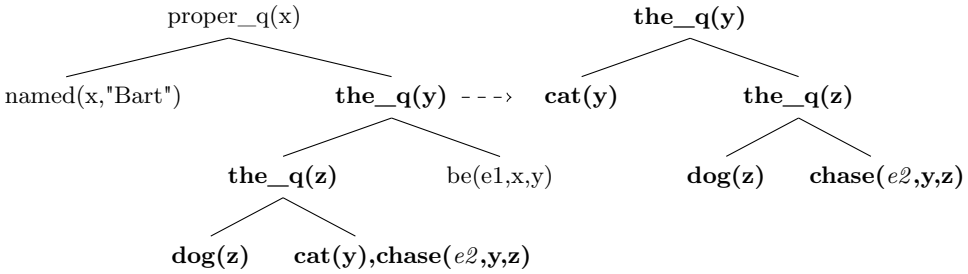


Fig. 1: The pipeline of a semantics-based question generation system.



(a) "Bart is the cat that chases the dog." → "Bart is the cat."



(b) "Bart is the cat that chases the dog." → "The cat chases the dog."

Fig. 2: A tree display of scoped MRS decomposition for subclauses. Upper-level tree nodes outscope lower-level nodes. The common suffix `_rel` is removed from all relations to save space. Scopes of each sentence is re-constructed after decomposition.

3.2 MRS Decomposition for Complex Sentences

A sentence that is too long generates lengthy questions, which is not desirable. Thus the semantic representations of complex sentences are first decomposed into partial and simpler ones. MrsQG employs four decomposers for apposition, coordination, subclause and subordinate clause. An extra WHY decomposer splits a causal sentence into two parts, reason and result, by extracting the arguments of the causal conjunction word, such as "because", "the reason", etc.

Fig. 2 shows an example of how the subclause decomposer works. Recall that the Davidsonian representation of MRS requires all verbs introduce events. Thus we assume every verb in a sentence represents an event, which is in the form of a sentence. The decomposer first identifies the event indicator (i.e. the verb[4]) in a sentence, such as "be" and "chase", then extract arguments of the verb and form new sentences for each verb. For instance, `be(e1,x,y)` takes two arguments, meaning "`x` is `y`". All EPs that take `x` and `y` as their bound variable are extracted and a new sentence "Bart is the cat" is assembled.

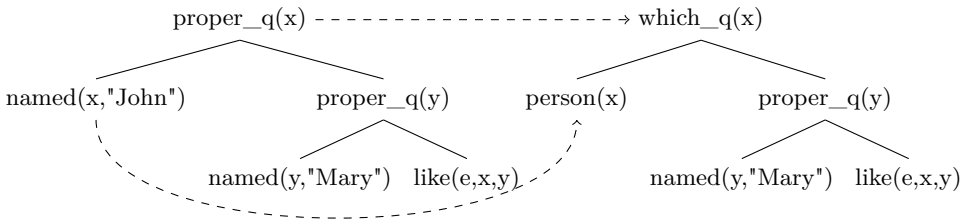
In the ontology of ERG, different linguistic structures are assign specific relations, such as `appos_rel` for apposition, `subord_rel` for subordinate clause. Thus very similar to the subclause decomposer, MRS decomposition spots these relations and extract their arguments to form new ones. In the current stage, whether the extracted semantic representation is true against the original one is not verified, which by itself is a research question for future investigation.

3.3 MRS Transformation for Simple Sentences

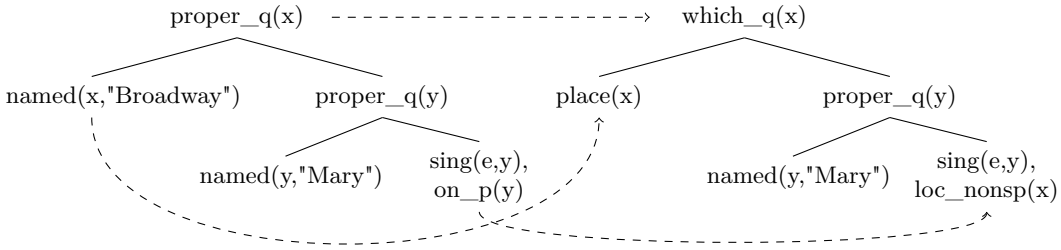
The transformation from declarative sentences into interrogatives follows a mapping between elementary predications (EPS) of relations. Fig. 3 has shown this mapping. Most terms in preprocessing are tagged as proper nouns (NNP or NNPS). Thus the EPS of a term turns out to consist of two EPs: `proper_q_rel` (a quantification relation) and `named_rel` (a naming relation), both of which have the same bound variable while `proper_q_rel` outscopes `named_rel`. The EPS of WH-question words have a similar parallel. For instance, the EPS of "who" consists of two relations: `which_q_rel` and `person_rel`, both of which also have the same bound variable while `which_q_rel` outscopes `person_rel`. Changing the EPS of terms to EPS of WH-question words naturally results in an MRS for WH-questions.

Similarly, in WHERE/WHEN/WHY questions, the EPS for the question word are `which_q_rel` and `place_rel/time_rel/reason_rel`. Special attentions must be paid to the preposition word that usually comes before location/time. A preposition word stays on the same node of a semantic tree as the head of the phrase this PP is attached to (as shown in Fig. 3(bcd)). The EP of the preposition must be changed to a `loc_nonsp_rel` EP (an implicit locative which does not specify a preposition) which takes the WH word relation as an argument in both cases of WHEN/WHERE. This EP avoids generating non-grammatical question words such as "in where" and "on when".

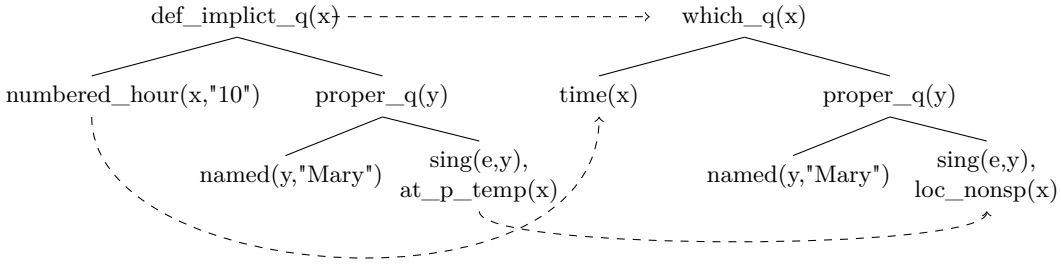
[4] The actual implementation is more complicated because there are other words such as non-scopal adverbs that also take events as bound variables.



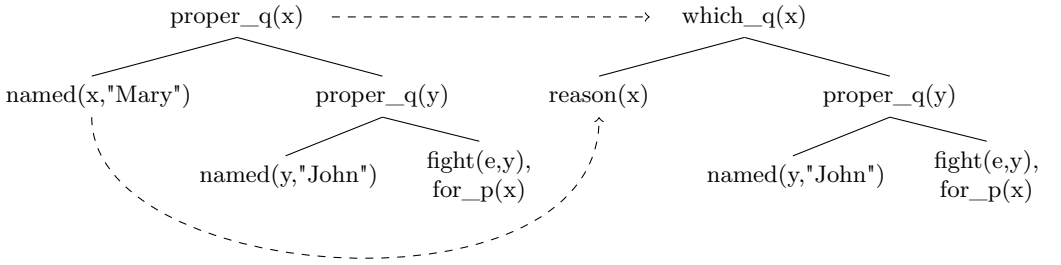
(a) "John likes Mary" → "Who likes Mary?"



(b) "Mary sings on Broadway." → "Where does Mary sing?"



(c) "Mary sings at 10." → "When does Mary sing?"



(d) "John fights for Mary." → "Why does John fight?"

Fig. 3: A tree display of scoped MRS relations and corresponding interrogative forms. Upper-level tree nodes outscope lower-level nodes. Some EPs are simplified to save space.

3.4 Language Independence and Domain Adaptability

MrsQG aims to stay language-neutral based on a semantics transformation of sentences. In principle, it needs little modification to adapt to other languages, as long as there is a grammar[5] conforming with the HPSG structure and LKB. However, the experience on multi-lingual grammar engineering has shown that although MRS offers a higher level of abstraction from the syntax, it is difficult to guarantee absolute language independence. As a syntax-semantics interface, part of the MRS representation will inevitably carry some language specificity. As a consequence, the MRS transfer needs to be adapted for the specific grammars, similar to the situations in MRS-based machine translation ([10]).

The domain adaptability is confined in the following parts:

1. Named entity recognizers (NER). For a different domain, the Stanford NER must be re-trained. MrsQG also uses an ontology NER. Thus collections of domain-specific named entities can be easily plugged-in to MrsQG.
2. HPSG parser. The PET parser needs to be re-trained on a new domain with an HPSG treebank. However, since the underlying HPSG grammars are mainly hand-written, they normally generalize well and have a steady performance on different domains.

4 Conclusion

We report on MrsQG, a semantics-based question generation prototype system participated in the the Question Generation Shared Task Evaluation Challenge 2010. The core technology of the system is built upon the idea of MRS-transfer. The system involves heavy machinery, including various preprocessing, parsing and generation with a linguistically deep grammar, and MRS rewriting. To our knowledge, this is the first implementation of a working system for the question generation task following the idea of [13]. The system is theoretically interesting in that it involves a chain of deep processing steps. With the help of a precision grammar (ERG), the grammaticality of the generation outputs is guaranteed. The manipulation on the semantic structure also allows one to produce various meaningful questions.

The development of the system also shows that there are various challenging research and engineering questions. Particularly, the MRS transfer component turns out to be fragile (i.e. a slightly ill-formed MRS will lead to generation failure). Also, the ranking of the generation outputs can be further improved by incorporating language models trained on large corpora. The multi-linguality and cross-domain applicability of the approach needs to be investigated in future research.

Acknowledgement We thank for discussions and help from Dr. Dan Flickinger and the DELPH-IN community. The first author acknowledges financial support

[5] For a list of available grammars, check `http://wiki.delph-in.net/moin/MatrixTop`

from the Erasmus Mundus Program through scholarships for the European Masters Program in Language and Communication Technologies (LCT). The second author thanks the German Excellence Cluster of Multimodal Computing and Interaction for the support of the work.

References

1. Ulrich Callmeier. PET – a platform for experimentation with efficient HPSG processing techniques. *Nat. Lang. Eng.*, 6(1), 2000.
2. J. Carroll and S. Oepen. High efficiency realization for a wide-coverage unification grammar. *Lecture notes in computer science*, 3651:165, 2005.
3. John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99)*, Toulouse, France, 1999.
4. A. Copestake, D. Flickinger, C. Pollard, and I.A. Sag. Minimal recursion semantics: An introduction. *Research on Language & Computation*, 3(4):281–332, 2005.
5. Ann Copestake. *Implementing Typed Feature Structure Grammars*. CSLI: Stanford, 2002.
6. Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, Morristown, NJ, USA, 2005.
7. Dan Flickinger. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28, 2000.
8. M. Heilman and N. A. Smith. Question Generation via Overgenerating Transformations and Ranking. Technical report, Language Technologies Institute, Carnegie Mellon University Technical Report CMU-LTI-09-013, 2009.
9. Jack Mostow and Wei Chen. Generating instruction automatically for the reading strategy of self-questioning. In *Proceeding of the 2009 conference on Artificial Intelligence in Education*, Amsterdam, The Netherlands, 2009.
10. Stephan Oepen, Helge Dyvik, Jan Tore Lønning, Erik Velldal, Dorothee Beermann, John Carroll, Dan Flickinger, Lars Hellan, Janne Bondi Johannessen, Paul Meurer, Torbjørn Nordgård, and Victoria Rosén. Som å kapp-ete med trollet? Towards MRS-based Norwegian-English Machine Translation. In *Proceedings of the 10th International Conference on Theoretical and Methodological Issues in Machine Translation*, Baltimore, MD, October 2004.
11. C.J. Pollard and I.A. Sag. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, 1994.
12. V. Rus and A. Graesser, editors. *The Question Generation Shared Task and Evaluation Challenge*. 2009.
13. Ivan A. Sag and Dan Flickinger. Generating questions with deep reversible grammars. In *Proceedings of the First Workshop on the Question Generation Shared Task and Evaluation Challenge*, Arlington, VA: NSF, 2008.
14. Brendan Wyse and Paul Piwek. Generating Questions from OpenLearn study units. In *Proceedings of the 2nd Workshop on Question Generation In Craig, S.D. & Dicheva, S. (Eds.) (2009) AIED 2009: 14th International Conference on Artificial Intelligence in Education: Workshops Proceedings*, 2009.
15. Yi Zhang, Stephan Oepen, and John Carroll. Efficiency in unification-based N-best parsing. In *IWPT '07: Proceedings of the 10th International Conference on Parsing Technologies*, pages 48–59, Morristown, NJ, USA, 2007.

QGSTEC System Description – JUQGG: A Rule based approach

Santanu Pal, Tapabrata Mondal, Partha Pakray, Dipankar Das and Sivaji Bandyopadhyay

Department of Computer Science and Engineering, Jadavpur University, India
{santanupersonal1, tapabratamondal, parthapakray, dipankar.dipnil2005}@gmail.com, sivaji_cse_ju@yahoo.com

Abstract. The article presents the experiments carried out as part of the participation in the Task-B of Question Generation Challenge, 2010. In the present task, generating questions from sentences requires preprocessing of the corpus with the additional knowledge of parsing, Semantic Role Labeling (SRL), Named Entities (NE) and causal *relaters*. The clause boundary detection in each sentence is carried out from parsed dependency relations based on the punctuation marks and verb specific heuristics. The identification of the cue phrase in each clause, replacement of the cue phrase with appropriate question word and reordering of the remaining chunks in the clause generate the question template. Evaluation of the generated questions through human rating on the development set gives satisfactory results.

Keywords: Question Generation, SRL, Parsing, Clause, Cue phrase.

1 Introduction

Question Generation (QG), the promising research arena of natural language generation and understanding, is defined as a task with simple or complex or compound sentences as input and the generated question as output [3].

The present work deals with the methodology to generate questions focusing on *person* Named Entities (NE), *temporal* or *location* information, *agent* based semantic roles associated with the words in the input sentences. A combinator module integrates the above information in each sentence and also tags the keywords in the sentence using a keyword list of causal verbs, negation words and numbers in words, both ordinal and cardinals. The combinatory module works with the dependency relations and part of speech tags as generated by the dependency parser. The chunker and clause detector module works with the output of the dependency parser and the root forms of the noun and verb words obtained from the Morphological analyzer available with the English WordNet [1]. The clause boundaries in each sentence are identified using the punctuation marks along with verb based heuristics (e.g. each verb refers to a clause associated with other subjective components identified through direct dependency relations). The question recognizer module checks whether a specific question type can be generated from a clause in each input sentence using the

output from combinator module and also identifies the possible cue phrase in the clause for the specified question type. The question recognizer module then forwards the identified clause to the question generator module. The generator module replaces the cue phrase by the question word and reorders the chunks in the clause to ensure grammatical correctness.

2 System Framework

The system architecture is shown in Figure 1.

2.1 Combinator Module

The input sentences are passed through the open source tools such as Semantic Role Labeler (SRL)[1] and Stanford Dependency Parser [2]. The keyword lists include the causal words, negation words and another list of ordinal and cardinal numbers. The *combinator* module identifies the person, temporal and location information from the semantic role labels. This module integrates all the tags in the input sentence and forwards it to the Question Recognizer module.

2.2 Chunker and Clause Detector

The chunking is done using a Conditional Random Field (CRF) based chunker [4]. The root forms of nouns and verbs are identified using the Morphological Analyzer available with the English WordNet [1]. The punctuation marks, discourse markers identified through *mark* () type dependency relations, causal words (*as*, *because*) are used for clause detection. The dependency relations connected directly with each verb chunk are used in clause detection. Each verb chunk and the associated chunks whose head is directly linked with the verb chunk in any dependency relation identifies a clause. The clause detection process helps to generates multiple questions of different types from a single sentence.

2.3 Question Recognizer (QR) Module

Based on the specified question type (Q-Type) and the tagged input sentence as obtained from the combinator module, this module identifies the cue phrases that are to be replaced by the appropriate question words in the question generator module. More than one cue phrase may be identified for a specified question type and hence more than one question of the specified type may be generated.

In case of *who* type questions, if SRL labels any phrase as first argument (tagged as *ARG0*) and if any word of that phrase is tagged as NNP and present in *nsubj*() dependency relation, the phrase is considered to be a possible question cue phrase for

[1] http://sourceforge.net/projects/swirl-parser/files/

framing *who* type questions. On the other hand, cue phrase for *what* type questions are generated based on any phrase that is labeled as *ARG0* provided that it respective clause contains both *ARG0* and *ARG1* labeled by SRL. Similarly, *when* type question cue phrases are identified if SRL labels any phrase as *ARGM-TMP* or *TEMPORAL* and the chunk of the phrase contains any word with POS category 'CD' and any word o *prep_in* () or prep_*since* () dependency relations. In casefof *where* type questions, if SRL labels any phrase as *ARGM-LOC* or *LOCATION* and the chunk of the phrase contains any word of POS category NN/NNP and if any word of that phrase is associated in *prep_of* () dependency relation, the phrase is marked as question cue phrase.If any extracted chunk contains multiple noun words that are also present in *nn* () dependency relation, the chunk is considered for *which* question cue phrase. The *prep* () type dependency relations are considered for generating special types of w*hich* questions (e.g *For which*, *In which* etc). A slightly different approach is considered for generating "*how many*", "*why*" and "*yes/no*" questions. The keyword lists are used to identify the possible question cue phrases of the three question types. A separate knowledgebase is prepared for generating different forms of *how* questions (e.g. *how many*, *how much*, *how old*, *what percentage of* etc.) after analyzing the development set. If any chunk contains any word with POS category CD and the phrase is not labeled as *ARGM-TMP* or *TEMPORAL* by SRL, the chunk is considered as the question cue phrase. The key words like *as*, *because* signifies the presence of causal item in a sentence and thus give the clues for generating *why* type question. Any chunk containing negation word (e.g. *not*, *no*) as well as any auxiliary verb is the cue phrase for generating *yes/no* questions. An example of question generation is as follows.

Text: Nash began work on the designs in 1815, and the Pavilion was completed in 1823.
Q-Type: **Who, *When***
SRL output with [***cue phrase***]: [**ARG0** Nash] [TARGET began] [ARG1 work on the designs] [***ARGM-TMP*** *in 1815*] and [ARG1 the Pavilion] was [TARGET completed] [***ARGM-TMP*** *in 1823*]
Generated Questions:
1. **Who** began work on the designs in 1815?
2. **W*hen*** Nash began work on the designs?
3. ***When*** *was* Pavilion completed in?

2.3 Question Generator (QG) Module

The module replaces the cue phrase in the clause by the specified question word. The reordering module generates possible questions from the respective clause by reordering the chunks in the clause based on rule based grammatical knowledge.

3 Evaluation

Evaluation is carried out on the development set of 81 sentences. The preliminary human evaluation suggests that, the question types of *when, how many, yes-no, what* and *why* gives satisfactory results. Different forms of *how* questions (e.g. *what percentage of*, *how much* etc) are generated. The *causal relaters* are used to generate the *why* type questions. The clause detection using the semantic role labels helps in filtering the *adjuncts* to frame questions mostly from the argument portions of the input sentences. The prepositional chunk attachment problem needs proper handling to improve the quality of the generated questions.

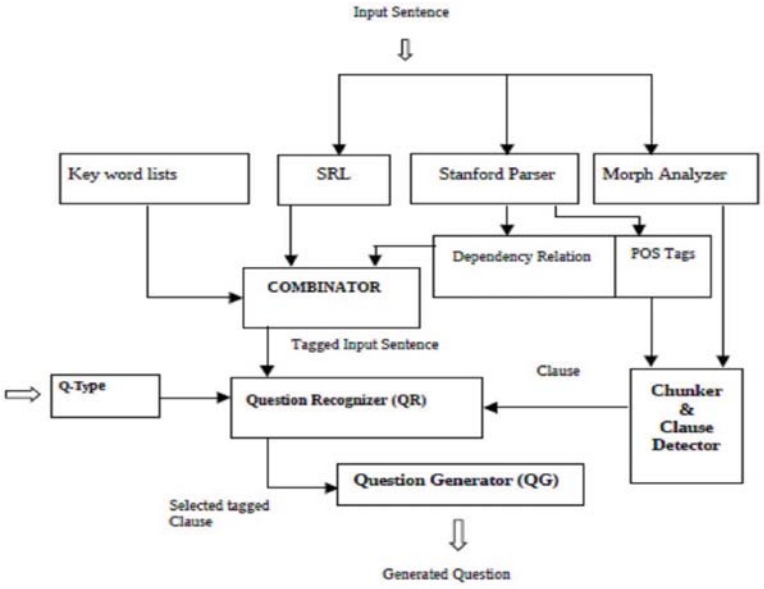


Fig. 1. Component based Conceptual System Diagram

References

1. George A. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312 (1990)
2. Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. *LREC* (2006)
3. Rus, V, Cai, Z and Graesser, AC. Evaluation in Natural Language Generation: The Question Generation Task. *Shared Tasks and Comparative Evaluation in NLG* (2007)
4. Xuan-Hieu Phan.: CRFChunker: CRF English Phrase Chunker. PACLIC 2006. (2006)

WLV: A Question Generation System for the QGSTEC 2010 Task B

Andrea Varga and Le An Ha

University of Wolverhampton
Research Group in Computational Linguistics
Research Institute in Information and Language Processing
Stafford Street, Wolverhampton, WV1 1SB, United Kingdom
{andrea.varga,l.a.ha}@wlv.ac.uk

Abstract. This paper presents the question generation system used by the University of Wolverhampton in QGSTEC 2010 task B. We have modified our multiple-choice question generation (MCQ) system in order to generate a new type of questions as requested by this task. We have also removed several constraints from our original system in order to generate more questions. In this paper, we will describe our approach and manual evaluation results on the development data.

Keywords: question generation, evaluation

1 Introduction

The solution adapted by the University of Wolverhampton to solve the QGSTEC 2010 task B relies on the methodology employed by the multiple-choice question generation system developed by Mitkov et al. ([1],[2]). The system is built on separate components, which perform the following tasks: (i) term extraction, (ii) question generation and (iii) distractor selection (which is not relevant in this task, and will not be discussed further). After key concepts/terms in a text have been identified, sentences containing these terms will be considered by the stem generation component. The stem generation component will check whether a sentence is suitable for generating questions. The criteria used include whether the terms occur in main clauses or subordinate clauses, whether the sentence has coordination structures, whether the sentence contains negative statements, etc. The main purpose of these checks is to ensure the quality of the generated questions (in other words, we try to generate fewer, but higher quality questions). After running the system on the data provided by QGSTEC, it becomes clear to us that some of these constraints need to be relaxed, and some modifications will be required in order to generate those target questions. The remainder of the paper describes our effort and evaluation, and is structured as follows: Section 2 presents the method used in this paper. Section 3 presents the evaluation on a development set. Finally, conclusions are drawn.

2 Method

In QGSTEC 2010 task B we extended our multiple-choice question generation system ([1]) by identifying the key phrases of the input sentences and implementing a few more syntactic rules for the missing question types.

The extended system follows the methodology of the initial system, and focuses on question generation, which employs various NLP techniques such as shallow parsing, key phrase identification, sentence transformation, and making use of language resources such as corpora, WordNet and Wikipedia. The system is built on several components, performing the following tasks: (i) identification of key phrases, (ii) assignment of semantic type, (iii) identification of question type and (iv) question generation.

Having a single sentence as input for question generation, the initial question generation system was extended by identifying all the key phrases in a given sentence using the Charniak parser ([4]). Noun phrases (NPs) satisfying the regular expression [AN]+N or [AN]*NP[AN]*N, as well as other phrases including prepositional phrases (e.g. in 1996) and adverbial phrases (e.g. early in the twentieth century) were considered as key phrases.

For the head of each phrase, a named entity recognition (NER) module assigns a semantic type (such as location, person, time, number and others). This NER module relies on information from a gazetteer, Wordnet, and Wikipedia. Based on the semantic type, the system decides the question type and the syntactic rule used to generate questions.

2.1 Syntactic Rules by Question Types

The question generation module takes as input source clauses, which are finite, contain at least one key phrase, and are of subject-verb-object (SVO) or SV structure. The following sections discuss the syntactic rules applied for the similar question types.

Which and What Questions. The question generation rules for the which question type are the same as in our initial multiple-choice question generation system. Namely, for SVO clauses containing a key phrase in subject position we create "WhichH VO" where "WhichH" is the subject of the clause in which the key phrase is replaced by "Which" and either the head of the NP (in the case of multi-word phrases), or a hypernym of the word retrieved from WordNet (in the case of single-word phrases). For key phrases that are in object position, depending on whether the V component contains any auxiliary verb, we front the auxiliary verb to construct the question or we use do-support. The same rules were applied for what questions, but instead of using "WhichH" as the question wording, we use "What".

Who, Whose, and Whom Questions. For noun phrases that have been recognised by the NER module as person names, instead of using "WhichH"

as the question wording, we use "Who" for NPs in subject position, "Whose" for NPs in possessive structure (e.g. Ian Fleming's novels) and "Whom" in any other case.

When and Where Questions. Prepositional and noun phrases are considered as key phrases for the where question type. The semantic label assigned by the NER module includes cities (in Leeds and the surrounding area), highlands (in the Scottish Highlands), capitals (Edinburgh), continents (into Europe) as well as countries (in Scotland).

The temporal expression recognition module ([3]) captures temporal expressions such as years (1996), months (May), centuries (the twentieth century), dates (October 4, 1951), references to present (now), durations (six year), special days (St Stephen's Day). The extent of these temporal expressions (noun phrases (fifteen years old), prepositional phrases (in 1996), adverbial phrases (early in the twentieth century)) serve as key temporal phrases of the sentence.

For these kinds of questions two new syntactic rules were implemented. These rules are modified versions of the original rules used to generate which questions. The first one fronts the main verb of the source clause and copies the content of the remaining clause except the key phrase. The second one replaces the content of the key phrase with the question word. For both rules we also ignore the subordinate clauses introduced by the "when" lexical trigger for the when questions (e.g. A buffer overflow occurs *when data written to a buffer, due to insufficient bounds checking, corrupts data values in memory addresses adjacent to the allocated buffer.*) and those introduced by the "where" lexical trigger for the where questions (e.g. According to the Biblical book of Daniel, at a young age Daniel was carried off to Babylon *where he was trained in the service of the court under the authority of Ashpenaz.*).

Why Questions. The syntactic rule applied for the why questions starts with the question word, fronts the main verb of the clause and copies the rest of the subclause ignoring the key phrase and subordinate clauses introduced by the "because" lexical trigger (e.g. A lot of very productive people get sick and/or die *because of smoking related diseases.*).

How Many Questions. Noun phrases that contain numeric expressions are recognised by the system, and used for generating questions. Instead of using the "WhichH" as the question wording, we use "How many H". If the numeric expressions are percentage ones, we use "How many percent..".

3 Evaluation on the Development Data

Although the development dataset suggests that 180 questions should be generated, the system was only able to generate 115. This is because we have not been able to build a model to generate yes/no questions, as well as the fact that the

transformation rules are not able to deal with sentences that are too complicated. These questions have been rated by two humans, who were both involved in the development of the system but tried to be objective. We focus on two measures: Relevance (measuring how well the question can be answered based on the input sentence) and Syntactic Correctness and Fluency (measuring the grammaticality of the sentences), as it is not clear to us how to score the Question type measure. The Ambiguity measure is also not clear (e.g. if the question is not syntactically correct and/or not fluent, it is difficult to say whether it is ambiguous). The inter annotation kappa agreement on the Relevance of the questions was 0.21 and on the Syntactic Correctness and Fluency was 0.22. We believe more times will be needed to study and extend the guidelines in order to get better agreement. With regard to the average values, for Human One, the average values of Relevance were 2.45 when taking into account the not-generated questions, and 1.57 when only taking into account the generated questions. For Human Two, the corresponding values were 2.85 and 2.2. For Syntactic Correctness and Fluency, the corresponding values were 2.85 and 2.2 for Human One, and 3.1 and 2.64 for Human Two.

Most of the errors made by the system were due to incorrectly parsed sentences and the failure to identify any source clause for a key phrase (e.g. In 1980 , the son of Vincent J. McMahon , Vincent Kennedy McMahon , founded Titan Sports , Inc. and in 1982 purchased *Capitol Wrestling Corporation* from his father .).

4 Conclusion

This paper has presented our participation in the QGSTEC 2010 task B with a question generation system. The system generated 115 questions out of the target of 180 questions for the different question types (which/what/who/when/where/ how many). The generated questions do not score very well in both relevancy and syntactic correctness measures. The agreement between two human judges is quite low.

References

1. Mitkov, R., Ha, L. A. and Karamanis, N.: A computer-aided environment for generating multiple-choice test items. Natural Language Engineering 12(2). 177-194. Cambridge University Press. (2006)
2. Mitkov, R. and Ha, L.A.: Computer-Aided Generation of Multiple-Choice Tests. In Proceedings of the HLT-NAACL 2003 Workshop on Building Educational Applications Using Natural Language Processing, Edmonton, Canada, (2003)
3. Varga, A., Puscasu, G. and Orasan, C: Identification of temporal expressions in the domain of tourism. In Proceedings of Knowledge Engineering: Principles and Techniques Conference, Cluj-Napoca, Romania (2009)
4. Charniak E.: A Maximum-Entropy-Inspired Parser. In Proceedings of the ANLP-NAACL 2000, USA. Morgan Kaufman Publishers, 2000.

Question Generation from Paragraphs at UPenn: QGSTEC System Description

Prashanth Mannem[§], Rashmi Prasad[†], and Aravind Joshi[†]

† University of Pennsylvania, Philadelphia, PA
rjprasad,joshi@seas.upenn.edu
§ International Institute of Information Technology, Hyderabad, India
prashanth@research.iiit.ac.in

Abstract. This paper describes the question generation system developed at UPenn for QGSTEC, 2010. The system uses predicate argument structures of sentences along with semantic roles for the question generation task from paragraphs. The semantic role labels are used to identify relevant parts of text before forming questions over them. The generated questions are then ranked to pick final six best questions.

Keywords: Question Generation, Question Generation from paragraphs, Semantic Role Labeling, Natural Language Processing

1 Introduction

The generation of questions from their declarative counterparts is a challenging task. The task gets even more difficult if one were to generate questions for the content in a paragraph instead of a sentence. The Question Generation Shared Task Evaluation Challenge 2010 (QGSTEC'10) organized in collaboration with the 3rd Workshop on Question Generation posed this problem as a challenge to interested participants [8].

This paper describes the question generation system developed at UPenn for the QGSTEC'10. We participated in the *QG from paragraphs* task. Given a set of paragraphs, the participants had to generate six questions from each paragraph with varying levels of scope. The paragraphs had on an average 8 sentences and were taken from Wikipedia, Yahoo answers, Open Learn and other similar sources. The full task description can be found at [9].

2 Approach

To generate proper questions out of a paragraph, the meaning of the sentences and the relations between them in the paragraph must be obtained. But capturing the exact true meaning of a paragraph is beyond the reach of current NLP systems. While discourse parsing is still at an early stage for it be employed in NLP applications, semantic role labeling (SRL) systems have achieved

reasonable accuracies (>80% F score) in providing shallow semantic analysis of individual sentences.

In this work, we use the semantic role labels (SRL parse) provided by ASSERT [2] extensively. For each sentence, ASSERT provides multiple predicate-argument structures for all the predicates in the sentence. Arguments include complements of the predicates as well as adjuncts. Use of SRL is attractive for QG because it identifies semantically meaningful parts for predicates in a sentence along with their semantic roles. These arguments can be used to ascertain whether a particular constituent is appropriate for generating a question or not.

Example 1: [*ARG1* `One-handed backhand players`] [*PRED* `move`] [*ARG2* `to the net`] [*ARGM-MNR* `with greater ease than two-handed players`] [*ARGM-CAU* `because the shot permits greater forward momentum and` $\cdots$]

In the above example, the ASSERT system identified `move` as the predicate with four arguments. `one-handed backhand players` is the *ARG1* and `to the net` is the *ARG2*. *ARGM-MNR* denotes the manner and *ARGM-CAU* indicates a causal argument. The semantic roles $ARG0 \cdots ARG5$ are mandatory arguments and the roles starting with *ARGM-* are the optional arguments [1]. We harness this information provided by the semantic role labeler for question generation.

The one thing that SRL does not handle is copular verbs, which are a common occurrence in the data and are also important for question generation. For this, we used information from the dependency parses and combined the result with the SRL parse.

Our QG system has three stages *content selection*, *question formation* and *ranking*. In the content selection stage, all the pieces of text in the paragraph over which the question has to be asked are identified. These pieces of text could either be a phrase or a clause or an entire sentence. In the second stage, a question is formed over each of the texts picked in the previous stage. The questions are ranked in the third stage using a heuristic function to pick the six best questions over the entire paragraph.

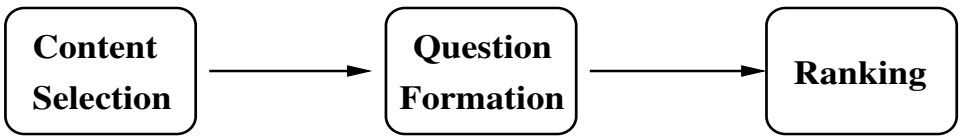


Fig. 1. Three Stages of QG system

3 Content Selection

Content selection is crucial for any natural language generation system. In case of question generation, it is the text over which the question has to be asked.

For a *general* question, the content is the entire paragraph. In case of a *medium* scope question, the content is typically spans 2-3 sentences in the paragraph. Whereas for *specific* questions it could be a phrase or a clause in a sentence.

As mentioned in Section 2, SRL parses are used for content selection. The ASSERT system is run over all the sentences in a paragraph to obtain the predicates, their semantic arguments and the semantic roles for the arguments. This information is used to identify the possible *target* content for a question. The **target** for a question is the text over which the question is to be asked. The *targets* are identified using the following selection criteria.

3.1 Mandatory arguments

Questions can be posed over the mandatory arguments of a predicate. A mandatory argument is an argument with one of the six semantic roles $\{ARG0 \cdots ARG5\}$.

For instance, in Example 1, the question could be formed over *ARG1* of `move`. `Who move to the net with greater ease than two-handed players?` (Ans: `one-handed backhand players`). On *ARG2*, the question could be `Where do one-handed backhand players move to with greater ease than two-handed players?` (Ans: `to the net`).

Mandatory arguments of all predicates in every sentence of the paragraph are considered as possible *targets* and passed over to the *question formation* stage. This method of identifying *targets* generates more than one *target* for a sentence and therefore a large number of *targets* for a paragraph. Predicates with less than two arguments are ignored since a question cannot be formed on just one argument.

3.2 Optional arguments

Certain types of optional arguments are informative, making them good *targets*. Arguments with roles $\{ARGM\text{-}MNR, ARGM\text{-}PNC, ARGM\text{-}CAU, ARGM\text{-}TMP, ARGM\text{-}LOC, ARGM\text{-}DIS\}$ are good candidates for being a *target*.

In Example 1, the question on *ARGM-CAU* of `move` would be `Why do one-handed backhand players move to the net with greater ease than two-handed players ?` Table 1 gives the description of these roles and the question type associated with each of the roles.

3.3 Copular verbs

The copular verbs are different to the other verbs because the semantic role labeler doesn't give predicate argument structure for them. We use the dependency parse to determine the arguments of a `be` verb. The right argument of the verb is almost always a good *target* for a question unless the sentence is existential (ex: `there is a` $\cdots$).

For instance, in the sentence `Latent semantic analysis (LSA) is a technique in natural language processing, in particular` $\cdots$, the right argument

Semantic role	Role description	Question type
ARGM-MNR	manner marker	How
ARGM-CAU	cause clause	Why
ARGM-PNC	purpose clause	Why
ARGM-TMP	temporal marker	When
ARGM-LOC	locative	Where
ARGM-DIS	discourse marker	How

Table 1. Optional args considered for question formation

of the verb `a technique in natural language processing` $\cdots$ is a good *target* for questioning. The question could be `What is Latent semantic analysis (LSA) ?` The left arguments for copular verbs are too complex to generate questions over. Hence, only the right arguments are passed as *targets* to the next stage.

4 Quesiton Formation

The *targets* along with the ASSERT analyses of the sentences are passed over from the previous stage to form questions on them. In this stage, the verb complex for each sentence is first identified and then a series of transformations are applied for every sentence corresponding to a *target* to generate the final question.

4.1 Identify Verb Complex

Verb complexes (VC) consist of the main verb along with any auxiliaries or modals (AUX) that might be present. For ex: `may be achieved`, `is removed` etc... The VC for each predicate corresponding to a *target* in the first stage, is identified from the dependency parse of the sentence. The identification of the verb complex is necessary for the syntactic transformations that need to be performed on a sentence to generate a question.

4.2 Transformations

Various transformations are applied on the sentence depending on the predicate and the *target* (its role, its relative position w.r.t the predicate). For *targets* which are either mandatory or optional arguments, syntactic transformations are performed in three steps.

– The first transformation involves deletion of the *target* identified in the earlier stage from the sentence and replacing it with the first preposition in the *target* (if there is any). For instance, for *target* [*ARG2* `to the net`] in Example 1, the sentence after this step would be

[*ARG1* ...] [*PRED* `move`] **to** [*ARGM-MNR*...] [*ARGM-PNC*...]

In case of the left *target* [*ARG1* `one-handed backhand players`], the sentence after the first transformation step would be

[*PRED* `move`] [*ARG1* ...] [*ARGM-MNR*...] [*ARGM-PNC*...]

- A *Wh* question word (`who, why, when, what, how, where`) is picked for the *target* depending on its semantic role and named entity tag. The default *Wh* word is `what`. Table 1 lists the question words for various optional arguments. `Who` is used if the argument has a named entity tag *PERSON* to its head word and `where` if the named entity is a *LOCATION*.
- In the third step, all the optional arguments to the left of the predicate are moved to the end of the sentence. Questions are well-formed when additional information is at the end rather than at the start of the sentence.
- Finally, syntactic transformation rules are applied on the verb complex of the sentence to transform the declarative sentence into a question. These rules look at whether or not the VC includes an AUX: if it does, the AUX is moved to initial position in the sentence, and the *Wh* word is inserted before the AUX; if the VC does not include an AUX, the mapping rules in Table 2 need to be applied to introduce an AUX for the question.

POS	AUX
VBD	did
VBP	do
VBZ	does

Table 2. Mapping rules for do-support in Questions

These mappings refer essentially to the POS of the main verb, which determines which AUX should be used for do-support in the question. This AUX is then inserted in initial position, the *Wh* word is inserted before the AUX, and finally, the main verb of the VC is converted to its lemmatized form.

In case of *targets* involving copular verbs, a similar approach is taken.

4.3 General Question

The first sentence of the paragraph is used to generate the *general* question. They undergo a different transformation process compared to the above method. If the sentence has a copular verb (most Wikipedia articles begin with it), the question if formed over the right argument as the *target*. If it doesn't, the question is formed by relativizing the right argument of the main clause of the sentence.

Example 1 [*ARG0*Intelligent tutoring systems] [*PRED* consist] [*ARG1* of four different subsystems or modules]
What are [the four different subsystems or modules] ***that*** *[intelligent tutoring systems] [consist] of ?*

Example 2 [*ARG0* But one-handed backhands] [*PRED* have] [*ARG1* some advantages over two-handed players]
What are [some advantages over two-handed players] ***that*** *[one-handed backhands] [have] ?*

4.4 Assigning Answer Scope and Context

The shared task requires the teams to not just generate questions, but also assign answer scope and context to these questions. The context is one of *specific*, *medium* and *general* while the answer scope is the span of the text in the paragraph that the answer to the question lies in.

We label each question as *general*, *medium*, and *specific* based on several criteria such as paragraph location, sentential syntax, and discourse connectives [3]. The questions that are generated from mandatory arguments are given a *specific* tag as context and the answer scope is the span of the argument. For optional arguments, depending on the role the context and answer scope changes. For the roles *ARGM-TMP* and *ARGM-LOC*, the context is always *specific* and the answer scope is the span of the argument. The adjunct semantic roles *ARGM-CAU* (causal) and *ARGM-PNC* (purpose) normally have scope beyond the clause, so we assign *medium* context to the questions generated on them. The role *ARGM-DIS* identifies few discourse markers like `as a result`, `consequently`. The context for questions involving *ARGM-DIS* is *medium* and the answer scope includes the current and previous sentences.

In our work, we assume that the question generated on the first sentence of the paragraph is the most general question. It is assigned a *general* context with the answer scope being the entire paragraph.

5 Ranking

While the system accumulates a list of *targets* after the content selection stage, it generates questions for these *targets* in the question formation stage. In the final stage, this set of questions is ranked to select the best 6 questions.

The list of questions are ranked in two steps. In the first step, the questions are ranked by the depth of the predicate in the dependency parse. Since the semantic role labeler gives predicate argument structures for all the predicates in the sentence, questions are also generated from unimportant predicates in complex long sentences. This ranking method makes sure that the questions generated from main clauses get a higher rank than the ones generated from subordinate clauses.

In the second step, the questions with same rank are ordered by the number of pronouns occurring in the question. Since the system doesn't handle coreferences, questions containing pronouns are given a lower score.

Finally, 6 questions are selected from the ranked list based on the task requirements.

6 Tools

In this work, we used the following free tools available for download on the internet to get a variety of information. Figure 2 gives the flowchart of the information flow among the various tools used by our QG system.

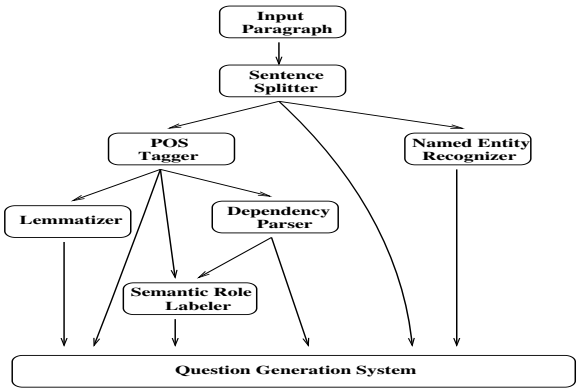


Fig. 2. Preprocessing Flow Chart

- *Sentence splitter*: **BART** coreference toolkit which also does sentence splitting was used for the task [7].
- *Lemmatizer*: The **MorphAdorner** by North Western University was used to get the lemmas for words in the paragraph [10].
- *POS tagger*: **Bidirectional POS tagger** using the bidirectional sequence labeling approach by Libin Shen [5]
- *Named entity tagger*: The named entity tagger recognises named entities in a given paragraph and classifies them into a predefined set of classes (like *person*,*organisation*). **Illinois Named Entity Tagger (LBJ based)** was used for this task [4].
- *Parser*: **Bidirectional LTAG dependency parser** by Libin Shen was used for this purpose [6].
- *Semantic role labeler*: The **ASSERT** toolkit by Sameer Pradhan and group was used [2].

7 Conclusion and Future Work

Our approach to question generation from paragraphs uses predicate argument structures to select content for question formation and then applies a series of transformations to generate a list of questions. A two step heuristic ranking picks the best six questions for the paragraph. This work is a proof-of-concept of the role of SRL in question generation and much work needs to be done on improving

the quality of questions. The tools used in our work are not 100% accurate particularly the semantic role labeler (<85% F score). Errors in ASSERT's output affected the overall quality of the system. In the development data, paragraphs from Wikipedia and Open Learn were easier to work with than the ones from Yahoo answers. The paragraphs from Yahoo answers were addressed in first and second person. Since all the tools for English are trained on the Penn Treebank (WSJ corpus), they performed poorly on those paragraphs, which indirectly created problems for the question generation.

Future effort can be put into correcting the errors in the transformations and handling various kinds of popular constructions. Discourse connectives are crucial and can be used to generate medium scope questions effectively.

References

1. Palmer, M., Gildea, D., Kingsbury, P. : The Proposition Bank: An Annotated Corpus of Semantic Roles. Computational Linguistics, 31:1., pp. 71-105 (2005)
2. Pradhan, S., Ward, W., Hacioglu, K., Martin, J., Jurafsky, D. : Shallow Semantic Parsing using Support Vector Machines. In: Proceedings of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics annual meeting (2004)
3. Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L.,Joshi, A., Webber, B. : The Penn Discourse TreeBank 2.0.. In: proceedings of 6th International Conference on Language Resources and Evaluation (2008)
4. Ratinov, L., Roth, D.: Design Challenges and Misconceptions in Named Entity Recognition. In: Proceedings of the Annual Conference on Computational Natural Language Learning (2009)
5. Shen, L., Satta, G., Joshi, A.: Guided Learning for Bidirectional Sequence Classification. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (2007)
6. Shen, L., Joshi, A.: LTAG Dependency Parsing with Bidirectional Incremental Construction. In: Proceedings of Conference on Empirical Methods in Natural Language Processing (2008)
7. Versley, Y., Ponzetto, S.P., Poesio, M., Eidelman, V., Jern, A., Smith, J., Yang, X., Moschitti, A.: BART: A Modular Toolkit for Coreference Resolution. In: Proceedings of the 6th International Conference on Language Resources and Evaluation (2008)
8. Question Generation Shared Task Evaluation Challenge, 2010, `http://www.questiongeneration.org/QGSTEC2010`
9. Open Question Generation from Paragraphs: Task Description `http://www.questiongeneration.org/QGSTEC2010/uploads/QG-fromParagraphs.doc`
10. MorphAdorner tool at North Western Univeristy, `http://morphadorner.northwestern.edu/`