2015

Utilization of common human queries in ranking automatically generated questions

Golestanirad, Sina

Lethbridge, Alta : University of Lethbridge, Dept. of Mathematics and Computer Science

UTILIZATION OF COMMON HUMAN QUERIES IN RANKING AUTOMATICALLY GENERATED QUESTIONS

SINA GOLESTANIRAD
Bachelor of Information Technologies, European Regional Academy, 2012

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

UTILIZATION OF COMMON HUMAN QUERIES IN RANKING AUTOMATICALLY GENERATED QUESTIONS

SINA GOLESTANIRAD

Date of Defense: November 26 , 2015

Dr. Yllias Chali Supervisor	Professor	PhD
Dr. Wendy Osborn Committee Member	Associate Professor	PhD
Dr. John Zhang Committee Member	Associate Professor	PhD
Dr. Howard Cheng	Associate Chair	PhD

Dedication

This dissertation is dedicated to my brilliant and outrageously loving and supportive wife,

Fozhan Safarpour.

Abstract

We challenge a form of Paragraph-to-Question generation task. We propose a question generation system which can be used to generate questions from a body of text. Our goal is to rank the generated questions by using Community-based Question Answering systems to calculate the importance of the questions beside tree kernel functions to assess how grammatically correct they are. The main assumption that our project is based on is that each body of text is related to a topic of interest and it has comprehensive information about the topic.

Acknowledgments

I believe that I am able to finish this work because of continuous encouragement from my beloved family members, although they were thousands of miles away from me. I find no words to thank them for that.
I am also very grateful to my supervisor, Dr. Yllias Chali, who always inspired me to work hard and stay in the right track. I also thank my committee members Dr. Wendy Osborn and Dr. John Zhang for their valuable suggestions and guidance.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nowadays, people are becoming busier and involved in many different aspects of life. It is a fact that people need to prepare to deal with groups of problems and shortcomings that they have never experienced before. Consequently, it causes them to ask questions and search for appropriate answers, especially when they receive a list of answers instead of a specific answer.

There are a large number of sources where people can refer to investigate about their queries and needs. For example, in a university, groups of students and instructors can be considered as a good source in order to ask questions. The problem is that people are not accessible all in one place. Furthermore, time also plays an important role when someone wants to address her/his questions with human beings. Another recent source that is receiving an immense amount of attention is the Internet (e.g. websites, forums and other online areas), but this novel resource has its own flaws. Imagine that we are going to search online for the information that is required. A very first issue that arises about online information is the huge amount of close but not desired data that may be given instead of the specific information that we are seeking. As a result, it may cause us to lose track and waste time.

We have discovered that when people need to find something on the Internet they almost always first use search engines such as Google, Bing and Yahoo. Subsequently, after entering their queries they receive a long list of websites, blogs or any other sources that the search engine found related to their queries. At this point, the issue is that information may or may not be related to what users are looking for (Hasan, 2013). This is a serious issue as it may take them a long time to go through all of those similar documents and search for what they are seeking. The fact that the provided documents are mostly similar can even make this process much more confusing. In other words, the less different they are, the more difficult it will be to filter them to get to what is desired.

Human beings are very curious by nature, but studies show that they are not very good at asking questions about topics. They are often forgetful, which causes difficulties in expressing what is in their minds (Hasan, 2013). How can humans overcome this problem? Imagine that someone has bought a new Apple laptop and is immediately curious to find out what was the first logo for Apple Inc. He/she may ask other people, but of course people should be accessible at the time. As we mention earlier, there is another choice to receive the information, we may use a search engine to get the data by writing the query “Apple Logos” in the search box. As a result, we will be provided with a huge amount of data which may have the exact information that is needed. However, they may also include other information, such as who designed the logo, where it was designed or any other information that we are not interested in. One reason could be the query that is used. Our query is one of the most important clues that search engines have in order to figure out what they need to search for. The more informative it is, the more probable the provided information is what we need. Now the question is how to help users to make up better queries.

We believe that if before showing the list of websites, the search engine identifies and shows some suggested queries, we would benefit from this Question Generation system. It could help us to stay on a track where chances are much higher to get the desired data at the end. The use of suggestions enable us to use the right queries to gain what we are looking for. Suggestions could be questions like: *What is the logo of Apple Inc.? What was the first logo designed for Apple Inc.? Do we have any information about where Apple's logo was designed?* etc., (Hasan, 2013). In this thesis, we address the challenge of Generating Questions for Topics (GQT), which is motivated by the fact that people do not always obtain the desired results from search engines. In our Question Generation system we consider search engines' users' queries as topics and then try to generate related questions.

1.2 Overview of the Thesis

In the task of topic to question generation, we assume that for each topic (a user's query in a search engine) there is a body of text having useful and comprehensive information about the topic. Then, our goal is to generate a large question pool containing different questions related to the topic of interest. The next step is to rank and show the top-ranked questions to the search engines' users in order to help them to find what exactly they are looking for. We need to rank the questions because the number of generated questions could be too huge to be shown and we may want to show only top-ranked ones. With regard to this goal, we use the related text to create the question pool.

We generate the questions for a given topic in two steps. First, we tag the named entities in the topic and its associated body of text. A named entity is a phrase that clearly identifies one item from a set of other items that have similar attributes. Examples of named entities are first and last names, geographic locations, ages, addresses, phone numbers and companies' names. Then, we apply some general rules and generate a set of questions which are called basic questions. At this level the answers for the basic questions may not be in the text (from where they are generated) and the reason of generating them is to have more varieties of questions. Generating answers is outside the scope of our research and can be considered as an approach to other fields such as Intelligent Tutorials and Information

Retrieval (IR) systems. Second, we use predicates and their arguments from the sentences in the given body of text to generate another set of questions called specific questions, the answers to which can be found from the text.

At this point, because the number of generated questions may be too large, we need to rank them and show the top ones to the users. Now the question is that how to rank them, or in other words, what would give a question a higher or lower rank? There are different aspects according to which we can apply different criteria to the task of ranking. For example, imagine that we have a set of questions and want to show them to a group of people whose English is not very good. In this case we would benefit from a ranking method that would try to determine which questions have been written with easier language and simpler words. For another example, if we want to show them to children we may need to have a system which will filter questions on sensitive issues, such as sexual or violence issues. In the continuation of this thesis we will introduce the criteria that our ranking system is based on.

In order to rank the questions we take two steps. First, we investigate other questions being asked by people in Community-Based Question Answering (CQA) systems such as Yahoo! Answers to see how common our questions are. Second, we apply tree kernel functions in order to compute the syntactic similarity between each question and the text (from which the question is generated). This way, we can determine how grammatically correct our questions are. Then the questions are ranked by their commonality and grammatical correctness.

1.3 Reaserch Questions

Our goal is to help search engine users choose more informative queries. To do so, we generate as many questions as possible which are related to the users' interests, and then show the top-ranked ones as our suggestions. Now the main issue is how to generate those questions. To find the answer first we need to break it down into more specified issues. In this thesis we address the following research questions:

1. How to generate questions from tagged named entities in sentences ?
2. How to simplify complex sentences and break them into simpler ones to generate more accurate questions ?
3. How to generate questions from tagged predicates and arguments in sentences ?
4. How to assess the importance of generated questions ?
5. How to assess the grammatical correctness of generated questions ?

1.4 Resources

The resources used in this thesis are:

Named Entity Recognition to tag named entities (NE) in the topics and related body of texts in order to generate basic questions. The tagger used in this thesis is available at `http://cogcomp.cs.illinois.edu/page/software_view/NETagger`.

Basic Question Generation Roles to use the tagged named entities in the topic and related body of text to generate basic questions. This set of questions has been added to the question pool to increase the variety.

Sentence Simplification to generate more accurate questions. Sentences have to be first simplified. A simplified factual statement extractor model has been used to do the simplification part. The simplifier is available at `http://www.ark.cs.cmu.edu/mheilman/questions/`

Automatic Statistical SEmantic Role Tagger to parse sentences in the text semantically by tagging verb predicates and arguments in the sentences. The semantic role tagger used in this thesis is available at `http://cemantix.org/software/assert.html`

Specific Question Generation Roles to use the tagged predicates and arguments in the body of texts to generate specific questions.

Tree Kernel Function to calculate the syntactic similarity between generated questions and related texts in order to investigate how grammatically correct the generated questions are. To have more information about tree kernel refer to `http://disi.unitn.it/moschitti/Tree-Kernel.htm`

Community-Based Question Answering (CQA) systems to investigate how common and consequently how important the generated questions are.

1.5 Contributions

Although, there has been some research on question generation, our research differs from them in the following aspects:

1. To have a question pool with more variety we use a set of question templates to generate basic questions, whose answers may not be extracted from the text where the questions are generated.

2. We use a set of rules to turn tagged predicates and arguments in sentences into specific questions.

3. To perform the task of question ranking we benefit from a well-known community question answering system to find similar and well formed questions.

4. We also take care of grammatical correctness of the generated questions by comparing them with other sentences which have been written with acceptable grammars.

1.6 Summary

In this chapter we introduced the main idea behind our thesis which will be thoroughly explained and broken down into details in the following chapters. The next chapter will be dedicated to introducing some of the related works in the question generation area and we also provide some background information that is needed through other chapters. In Chapter 3 we will completely introduce and explain all steps of our question generation system and also the tools that we used to accomplish each step of the system. Chapter 4 is dedicated to breaking down and clarifying all components of our ranking algorithm beside acquainting readers with the database which is used in the algorithm. In Chapter 5 we illustrate our evaluation methodology, experiments and finally discuss the results.

Chapter 2

Background

2.1 Introduction

Recent studies reveal the fact that human beings are not very skillful at explaining their needs by asking informative questions (Heilman and Smith, 2010b). Therefore, a Question Generation system may be considered as a good approach to overcome this weakness by assisting humans to generate better inquiries. The Question Generation (QG) task, which is about generating a pool of reasonable questions from a given body of text, can be considered as a research topic that is receiving an immense amount of attention from researchers of different areas such as Natural Language Processing (NLP), Discourse and Dialogue, Natural Language Generation (NLG), Psycholinguistics, Intelligent Tutoring System and Information Retrieval (IR) (Rus et al., 2010).

In an intelligent tutoring system, a QG approach can be very useful to perform the task of generating questions from materials which should be learnt by students. At this point, generated questions can be used to assess the learners' accomplishments or even as an aid to help them to memorise or note the key aspects (Heilman and Smith, 2010b). Another example of the usage of a QG system is Complex Question Answering. A complex question usually consists of a series of simpler questions and they are often about a topic of interest. Therefore, by decomposing a complex question to a set of simple questions we will be able to use techniques which have been developed to find answers for simple questions. From another point of view, the task of complex question answering is closely related to the challenge of topic-to-question generation, because as we mentioned, complex questions are usually asking about a topic of the user's interest (Hasan, 2013).

2.2 Related Work

These days, Automated Question Generation systems are receiving much attention. There are a number of different methods to accomplish this task across different fields (Andrenucci and Sneiders, 2005). The Question Generation system that is most similar to ours has been created by Hasan (2013). He wanted to assist people who lack the ability to express the information that they are seeking. His system uses a title and a related text that has useful information about the title, and utilizes the information in the text to generate questions. Our approach is similar to the system proposed by Hasan (2013) because we use both predicate arguments, named entities and question roles in order to generate the questions. To assess the importance of the generated questions Hasan's system used Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to identify some subtopics in the text. Then, he applied Extended String Subsequence Kernel (ESSK) to assess the similarities between the subtopics and the generated questions. Because all of the questions are generated automatically, some flaws may happen. For example, the selected question words, such as What, Where and Who may not be suitable for the generated question. This is why syntactic correctness of generated questions should also be assessed. To do so, Hasan (2013) reimplemented the tree kernel model in order to calculate the syntactic similarities between each question and the text where it is generated from. In the research done by Hasan (2013) one of the criteria to rank the generated questions is their importance. Importance was estimated by determining similarities between the questions and the text. We believe that Community-Based Question Answering (CQA) systems are better sources of estimates to calculate the importance of the generated questions.

Many other efforts to accomplish the QG task are evident from the literature. Rokhlenko and Szpektor (2013) challenged the task of automatically generating questions, which are relevant to a given text but do not exist in the text. The motivation behind their approach is to increase users' engagement in news articles by generating some relevant questions from the news.

Kalady et al. (2010) presented an approach to question generation which is based on syntactic and keyword modeling. They used named entity recognition, parse tree manipulation and significant phrases in a document to generate definitional and factoid questions from documents. Kalady et al. (2010) generated different types of questions from a single sentence, including What, Where, Who, How and Yes-No questions. This factoid question generation system manipulates the individual sentences from a given document. At first, each sentence of the document is preprocessed. They parse the sentences and then execute pronoun replacement, and separate complicated sentences into some independent clauses. Next, each independent clause is passed into a question generation system, whose output is one or more questions from that clause.

Piwek and Stoyanchev (2010) addressed the task of creating dialogues from monologues. Their system converts statements into questions. They used QG to create more effective ways of presenting instructional content.

Automatic techniques for generating vocabulary questions have been addressed by researchers. For example, Brown et al. (2005) introduced a multiple-choice question generation method about lexical relationships (e.g. synonym and antonymy) using WordNet (Miller et al., 1990) as a source of lexical knowledge.

Mitkov and Ha (2003) created a system which uses a set of rules for generating questions from shallow parses and specific types of sentences (e.g., from a sentence with subject_verb_object order). They assessed their system by judging the quality of some questions which were generated from an online linguistics textbook.

Generally a Question Generation system (QG) consists of several steps. First, it is given a text from which the system selects one or more targets for questions ans specifies types of questions. Finally, when both targets and question types are determined, the system generates the questions (Hasan, 2013). In our proposed system we use named entities in addition to syntactic and semantic information in the sentences in order to generate the questions. Boyer and Piwek (2010) describe several approaches where named entities and/or syntactic and/or semantic information have been used to generate questions.

The main goal of the system that we are proposing is to automatically generate questions which are relevant to a query. To do this, our system uses predicate arguments and named entities in the body of the text to generate questions and then ranks them according to their importance and syntactic correctness. Our system generates some basic questions in order to increase the diversity of questions in our pool, though we do not have their corresponding answers. That is why our approach could be considered different in comparison with the setup offered in the Question Generation Shared Task and Evaluation Challenge (Rus and Arthur, 2009).

Another significant work in the area of Questions Generation has been proposed by Ali et al. (2010). He considered a Text-to-Question generation task using syntactic parsing, part of speech (POS) and named entity tagging. In his proposed system different modules have been used to process the raw data and to generate elementary sentences from complex sentences. Then these elementary sentences which are tagged by part of speech and named entity analyzers will be utilized to classify the sentences in order to determine the possible question types. Finally, the generated questions are ranked based on the grammatical structure of the elementary sentences.

Labutov et al. (2015) developed a system to generate more specific questions from texts that bypass many challenges of creating a full semantic representation. This is accomplished by decomposing the task into an ontology crowd-relevance workflow, which consists of representing the original text in a low-dimensional ontology, then crowdsourcing candidate question templates. Finally they rank potentially relevant templates for a novel text. In case that ontological labels are not available, they are inferred from the text.

Agarwal and Mannem (2011) recommended a system which automatically generates questions from a natural language text by using discourse connectives. Discourse connectives are words or phrases that connect or relate two coherent sentences or phrases and indicate the presence of discourse relations. They explore the usefulness of the discourse connectives for Question Generation (QG) systems. Their work classified the QG task into content selection and question formation. Content selection consists of finding the relevant part in text to frame questions. Question formation involves sense disambiguation of the discourse connectives, identifying the question types and applying syntactic transformations on the content. This system is evaluated manually for syntactic and semantic correctness.

As we mentioned earlier, our system extracts predicates and their arguments of the sentences in order to generate the second group of questions. Mannem et al. (2010) developed a question generation system at UPenn for QGSTEC, 2010. Their system exploits predicate argument structures of the sentences and also semantic roles for the question generation task from paragraphs. The semantic role labels are used to recognize the relevant parts of the text before generating the questions over them. At the end, the generated questions are ranked to pick the top six best questions.

There are many prior works on Question Generation which focused on the grammaticality of generated questions and generating effective multiple-choice distractors for individual questions. These days not all learning takes place in an educational setting. Self-motivated learners are becoming more interested in on-line areas to study about new topics (Becker et al., 2012). Becker et al. (2012) addressed the task of Question Generation to help such learners with the benefits of testing by automatically generating quiz questions for online documents. Their work concentrates on the aspect of determining which part of a given sentence should be used to ask about the topic in the first place. They address this problem by asking human judges to score the quality of generated questions from a Wikipedia-based corpus. Then, they train a model to effectively replicate these judgments.

Gap-fill questions are fill-in-the-blank questions, where one or more words are being missed from a sentence or a paragraph, and readers are provided with a list of potential answers. Gap-fill questions are easy to evaluate, but the issue is that preparing those questions manually is very time consuming and take a lot of effort. This is the point that automatic Gap-fill Question Generation (GFQG) from a given document can be very useful (Agarwal and Mannem, 2011). Agarwal and Mannem (2011) presented an Automatic Question Generation system that can create gap-fill questions for sentences in a given text. Their system collects the informative sentences from the document and constructs gap-fill questions from them. At first, they select and blank keys from the clauses and then determine the distractors for these keys.

2.2.1 Educational Purposes

An automated Question Generation system can also be used for educational purposes (Heilman and Smith, 2010b). Some papers address the task of automatically generating questions from reading materials in order to advance educational assessment and practice. Heilman and Smith (2010b) proposed a system that overgenerates a set of questions and then uses a model, which has been trained on a dataset in order to rank the generated questions. Liu et al. (2010) proposed an automatic question generation system which helps students to write literature reviews. Their system extracts and classifies citations from students' reviews, and then some templates are used to generate questions in order to provide needed feedbacks about missing content in the students' works. Gates (2008) developed a QG system to generate questions in order to help students while reading an article (rather than afterwards). This system tries to encourage students to look-back and re-read articles (e.g., if they do not know an answer). When instructors prepare learning materials for their students, they mostly construct accompanying questions to guide learning. Natural language processing technology can be used to automatically generate such questions (Lindberg et al., 2013). Lindberg et al. (2013) introduced a sophisticated template based system which merges semantic role labels into a system that automatically generates natural language questions to support online learning. While students read explanatory texts, their comprehension will be improved if they are paused by the system and asked to answer questions that reinforce the material (Mazidi and Nielsen, 2014). Mazidi and Nielsen (2014) proposed an automatic question generator which benefits from semantic pattern recognition to generate questions which have different depth and type for tutoring or self-study purposes. In their system, semantic role labels of the sentences in the source are used to generate both questions and answers related to the sentence in the source. Chen (2009) believes that good readers ask themselves questions while they are reading. His goal is to advance this self-questioning strategy automatically to help children in grades 1-3 understand informational text. He showed that instructions for self-questioning can be generated for narrative text.

2.3 Background Theory & Information

2.3.1 Latent Dirichlet Allocation

In our project, one of the key concepts is to calculate the semantic similarity between sentences (to be explained in detail in chapter 4). There are plenty of systems that perform the task of semantic similarity estimation between words, sentences and texts. We benefit from a set of similarity measures based on the unsupervised method Latent Dirichlet Allocation (Blei et al., 2003). In this section we will explain the main concept of the LDA model.

Let's assume that we have the following set of sentences:

1. I like to eat bananas and broccoli.
2. I ate some spinach smoothie and one banana for my breakfast.
3. Kittens and also chinchillas are cute.
4. My little sister adopted a kitten last week.
5. Look at that hamster munching on a piece of broccoli, it is cute.

What is Latent Dirichlet Allocation? It is a technique for automatically generating some topics that are contained in these sentences. For example, given these sentences and a request for two topics, LDA may generate something like:

Topic A: 30% broccoli, 15% bananas, 10% breakfast, 10% munching, ... (you may interpret topic A to be about food)
Topic B: 20% chinchillas, 20% kittens, 20% cute, 15% hamster, ... (you may interpret topic B to be about cute animals)
Sentences 1 and 2: 100% Topic A
Sentences 3 and 4: 100% Topic B
Sentence 5: 30% Topic A, 70% Topic B

The question is: how does LDA perform this discovery? In more detail, LDA displays documents as combination of topics which give out words with specific probabilities. It supposes that a document is generated in the following fashion. When you are writing a document, you

1. determine the number of words that the document will have.
2. pick a topic mixture for the document. For instance, if we have the two topics of food and cute animal from above, you may want the document to consist of 1/3 food and 2/3 cute animals.
3. generate each word in the document by:
   Picking a topic first, with regards to the multinomial distribution which is sampled above. For instance, you might choose the food topic with 1/3 probability and the cute animals topic with 2/3 probability, and then using the topic to generate the word itself. For example, if you picked the food topic, you would generate the word “bananas” with 15% probability, the word “broccoli” with 30% probability, and so on.

Assuming this generative model for a set of documents, LDA then make efforts to backtrack from the documents to find a set of topics which were likely used to generate the documents. Let us see a short example to see how exactly the LDA model works. According to the explanation given above, when you are going to generate a particular document named *doc*, you might

- Pick 4 (or any other numbers) to be the number of words in *doc*.
- Decide that *doc* will be 1/4 about food items and 3/4 about animals.
- Select the first word from the food topic, which then may give you the word “banana”.

- Select the second word from the animals topic, which may give you “hamster”.
- Select the third word from the animals topic, providing you with “cute”.
- Select the fourth from the animals topic, giving you the word “kittens”.

So the document that has been generated under our LDA model will be “ banana hamster cute kitten ” (we note that LDA is known as a bag-of-words model).

At this point, imagine that we have a set of documents. We have chosen some fixed number of K topics to be discovered, and need to use the LDA model in oder to learn the topic representation for each document and also the associated words for each topic. How can we peform this task? One way known as Collapsed Gibbs sampling, is the following:

- We must walk through each document, and randomly assign each word in that document to one of our K topics.
- We should notice that this way of random assignment already provide us both topic representations for all those documents and word distributions for all K topics, but albeit this is not a very good way.
- Therefore, if we want to improve them, for each document d we have to ...
  - Select each word *w*in the document *d*...
    - And compute two probabilities for each topic t: 1) p(topic *t* |document *d*) = the proportion of words existing in document *d* which are currently assigned to topic *t*, then 2) p(word *w* |topic *t*) = the proportion of assignments to topic *t* over all of the documents which come from this word *w*. After these two steps we need to reassign *w* to a new topic, where we select topic *t* with probability p(topic *t* |document *d*) * p(word *w* |topic *t*). According to the generative model, this is essentially the probability that topic t has been used to generate word *w*.

    * In other words, for this step, it is assumed that all topic assignments besides the current word in question are correct. Then, we update the assignment of the current word using the model of how the documents have been generated.

- Then all that is required is to repeat the previous step many times, until we reach a steady state where our assignments are acceptable. At this point, we count the proportion of words assigned to each topic within that document to estimate the topic mixtures of each document, and then the words associated to each topic.

2.3.2 Tree Kernel Functions

In the field of Natural Language Processing, it is usually important to compare parse trees to calculate their similarities. Tree Kernel (TK) functions can be used to do the task of computing the number of subtrees which are common between two parse trees. There is a constraint, which states that the subtrees nodes should be taken with none or all of the children that they have in the original tree. To measure the similarity between two syntactic trees, we represent each tree T by a vector having m dimentions, $v(T) = (v_1(T), v_2(T), ..., v_m(T))$, where $v_2(T)$ shows the number of occurrences of the second tree segment in tree T. The segments of a tree are all subtrees which contain at least one child. Figure 2.1 represents a tree with a portion of the subtrees. To calculate the tree kernel of two trees for example *T1* and *T2* we can calculate the inner product of those two trees' vectors:

$$TK(T_1, T_2) = v(T_1).v(T_2)$$

At this point we define an indicator function. Let $I_i(n)$ be the indicator, it will be set to 1 if the i_{th} subtree is rooted at node n, and 0 otherwise. It follows:

$$v_i(T_1) = \sum_{n_1 \in N_1} I_i(n_1)$$

$$v_i(T_2) = \sum_{n_2 \in N_2} I_i(n_2)$$

where, N1 and N2 are the set of nodes in T1 and T2 respectively. So, we can derive:

$$TK(T_1, T_2) = v(T_1).v(T_2) = \sum_i v_i(T_1) v_i(T_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) I_i(n_2)$$

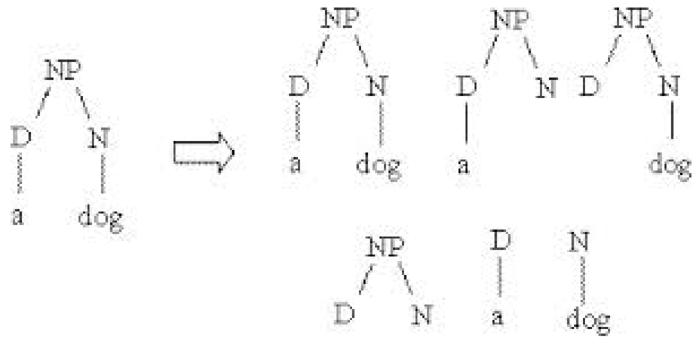

Figure 2.1: An example of a tree and its sub-trees

2.3.3 SEMILAR

In our experiments, we use the semantic similarity calculation toolkit SEMILAR[1] (Rus et al., 2013a). It has two main methods for calculating the semantic similarity scores, LDA-

[1]Available at http://www.semanticsimilarity.org/.

Optimal and LDA-Greedy. The main idea is that when it is given two sentences it calculates the semantic similarity between each pair of words in sentences. We use the LDA-Optimal method because unlike LDA-Greedy, this optimal method tries to find the best overall word to word match. There are also some other methods which uses the LSA model but unlike LDA they have less accuracy. (LDA) (Blei et al., 2003) creates a number of topics, each containing a set of words. A meaning or a concept (for example a word sense) is defined by each of these topics. LDA makes it possible that one word belongs to more than one topic, which means that different senses of the word are being covered by different topics.

This is why LDA is known to have a conceptual advantage over Latent Semantic Analysis (LSA) (Landauer, 2006). In LSA there is only one unique representation for each word. That is, all senses of one word are being represented by the same LSA vector, which makes it difficult to ascertain which meaning is being represented (Rus et al., 2013b). For example, LDA handles polysemy. A polyseme is a word or phrase with multiple meanings.

SEMILAR uses the Touchstone Applied Science Associates (TASA) corpus in order to train the LDA model. The TASA corpus has 60,527 samples which have been collected from 6,333 textbooks, popular works of fiction and nonfiction, and also works of literature. TASA consists of 154,941 word types and more than 17 million word tokens.

2.4 Summary

In this chapter we described the concept of Question Generation and showed some shortcomings that have motivated us and other researchers to work on different aspects of this area. We also explained some ideas which are related to our project or may give you an opinion of what other fields of Question Generation are about. In this chapter we introduced the idea of Latent Dirichlet Allocation as it is used in one of our key toolkits called SEMILAR (Rus et al., 2013a) to measure the semantic similarities (which will be describe in Chapter 4). Next chapter will be dedicated to Topic to Question Generation which is one of the Natural Language Processing fields that is receiving more attention recently. We will take you through all the steps of our system and explain them in all detail. At each section we will use an example to make it more understandable.

Chapter 3

Automatic Topic to Question Generation

3.1 Introduction

In this section we introduce Topic to Question Generation and give examples about its usage. Then, we narrow down to the exact case for which we are generating questions for a specific purpose.

When people need to gain required information about a topic, they usually start to ask questions about it. Imagine that a computer company has hired you as a tester to do Quality Assurance. There are many different programming languages and frameworks that you can use to do this task. Let assume that your company is using JAVA and SELENIUM and you have no idea about them. As your first step you start asking questions about these topics, but what type of questions? It has been noticed by researchers that usually a complex question is being used to ask about a topic of interest. A complex question consists of different aspects of a topic and can be broken down to a set of simple questions. For example, a complex question about our programming languages issue can be *what is the fastest and easiest way of learning JAVA and SELENIUM to use them in testing applications?* which could be broken to a group of simple questions such as, *what is the fastest way to learn JAVA? what is the fastest way to learn SELENIUM? how to use JAVA in testing? how to use SELENIUM in testing?* From another point of view, the task of complex question answering can be considered as the task of breaking a mixed question into simpler queries and then trying to find answers to each of them. In our case, in order to break down our complex question we can generate simple questions about JAVA and SELENIUM. That is why Topic to Question Generation can be used as a partial solution to the challenge of complex question answering, or more generally as a part of Question Answering (QA) systems.

There are some other areas where generating questions for a topic of interest could be applied. When people use one of the standard document retrieval systems, such as search engines they will be provided with a list of relevant documents. However, researchers believe that at this point the task of searching is not over as they have to persue all of those served (and usually ranked) documents to find the exact piece of information that they were looking for (Hasan, 2013). This method which makes users persue all relevant documents in order to gain the required information could be very time consuming. Another problem that may occur as a result of this searching method is to miss the correct answer easily by either inattentive reading or using an incorrect query. In case that an incorrect query is used, it may probably lead the retrieval system to provide the user with undesired documents. Retrieval systems such as search engines can be considered as QA systems. One of the key points in those systems is to receive well-formed queries as their inputs in order to be able to provide the best possible answer as their outputs.

Available studies show that humans usually lack the abilities to express their needs (Hasan, 2013). These days as people are getting busier they are becoming more forgetful and sometimes careless in using correct words and structures to generate informative questions. Well-structured and illustrative questions can be used to explain users' needs and consequently provide them with a set of correct answers. At this point, Topic to Question Generation can be considered as an acceptable procedure to overcome humans' weakness in generating questions. If we think of humans' queries as topics of interests they will benefit from a QG system which offers them some well-formed questions related to the topics of interest. This way retrieval systems will be collecting more accurate answers (documents) as they have received better structured and more explanatory inputs.

3.2 Task Definition

Our goal is to assist search engine users by creating a pool of well-formed and explanatory queries. To do so, we consider users' queries as topics of interest. This proposed system is based on an assumption that for each query (topic of interest) there exists a document from where we can extract comprehensive and related information. In other words, we propose an automatic question generation system which generates questions from a given body of text which is related to a topic of interest. Our question generation approach is built in six steps:

1. In the first step, we tag named entities from the text which is related to the query.
2. In the next step, we use question templates to generate basic questions, based on the tags from the previous step.
3. In the third step, we apply a text simplifier to all of the sentences in the text.
4. In the fourth step we use a semantic role tagger to tag all of the arguments and predicates in the simplified sentences.
5. The fifth step is about applying another set of question rules to the extracted arguments and predicates in order to generate specific questions.
6. In the final step we use our proposed algorithm to rank all of the generated questions.

3.3 Named Entity Recognition (NER) Tagger and Basic Questions

In this section we explain how to tag named entities, which is the first step of our question generation system. Named-Entity Recognition (NER), which is also referred to as Entity Extraction, Entity Chunking and Entity Identification, can be considered as a subtask of Information Extraction. The Information Extraction task is about searching, locating

and classifying elements in a given body of text into some predefined categories, for example, organizations, locations, the names of people, expressions of times, monetary values, percentages and quantities. Most NER systems are structured to take an unannotated piece of text, such as:
Sina bought 500 shares of Acme Corp. in 2016.
and construct an annotated text which highlights the entities:
Sina[Person] bought 500 shares of Acme Corp.[Organization] in 2016[Time].
As we can see, a two-token company name, a person's name and a temporal expression are identified and then classified.

The ability to recognize different types of entities and categories in a given body of text is required to understand the text to the level that we can exploit needed information from it. Then, we can use an intelligent way to answer or generate questions with respect to them. For instance, providing us with the information saying this phrase represents an organization, a location, a person's name or other semantic categories can be helpful to generate or answer questions about the phrase. Named-Entity Recognition is also a kind of context sensitive challenge. For example, ”Washington” is a city in one context and a person's name in another. Machine learning techniques have been used to deal with this problem and distinguish appropriate semantic categories for entities (Hasan, 2013).

In our project, the named entities, which are in the topics and their relevant texts are tagged using the Illinois Named Entity Tagger[2]. Then we apply our question templates (which can be found at Appendix A) to generate the basic questions. At this level, the answers for these questions may not be in the text and the reason for generating these questions is to have more diversity in our question pool. The NER tagger used by Hasan (2013) uses just four labels to tag the named entities:

[2]Available at http://cogcomp.cs.illinois.edu/page/software_view/NETagger.

- PERSON-Person
- ORG-Organization
- LOC-Location
- Misc-Miscellaneous

However, the NER tagger that is used in our proposed system uses 18 labels:

- PERSON-Person
- ORG-Organization
- LOC-Location
- TIME-Time
- LAW-Law
- NORP-Nationality
- GPE-Geopolitical Entity
- LANGUAGE-Language
- PERCENT-Percentage
- FAC-Facility
- PRODUCT-Product
- ORDINAL-Ordinal Number
- CARDINAL-Cardinal Number
- WORK_OF_ART-Work of Art
- MONEY-Money

- DATE-Date
- EVENT-Event
- QUANTITY-Quantity

3.4 Basic Question Generation

We designed 265 question templates (Which can be found at Appendix A) and a program written in JAVA that generates the basic questions with regard to the tagged named entities. There were some challenges while creating templates as our information about the tagged named entities is just about their types. For example, a noun could be tagged as a person's name and one of the good templates could ask about her/his death date, but what if that person is still alive? As you can see there are many good questions that can be asked about a specific topic but at the same time they may mislead the users. Another issue was with some of the tag label types. For instance, we had many numbers tagged as ordinal numbers but what could be asked about them when we have no other information? Therefore we should generate questions about those labels that are as general as possible. Other problems are caused by the shortcomings of the tagging toolkit. As an example, it usually happened that a person's name was tagged as a company name which could lead us to generate many meaningless questions. These issues forced us to decrease the number of templates in order to reduce the chance of generating unusual questions. Table 3.1, shows some examples of our templates and generated questions.

In the case of basic questions, the only clue that we are provided to generate questions are tagged named entities and no other information is given, so we can say answers to these questions may or may not be in the text from where they are extracted.

Table 3.1: Examples of templates to generate basic questions

Tag	**Named Entity**	Generated Question
FAC	iMac	When was *iMac* produced first ?
ORG	Greenpeace Foundation	How many people work at *Greenpeace Foundation* ?
PERSON	John Cormack	When was *John Cormack* born ?
GPE	Texas	Have there ever been any natural disasters in *Texas* ?
EVENT	Christmas	How much money is spent for *Christmas* ?
Work Of Art	Mona Lisa	Who created *Mona Lisa* ?

3.5 Sentence Simplification

Text simplification is an essential operation which has been used in different areas of Natural Language Processing. Text simplifiers are usually applied in order to modify and process an existing corpus of a human readable body of text in such a way that the structure and also the grammar of the text is substantially simplified, while the meaning and information in the text remain the same. The task of text simplification is a serious area of research, because natural human languages usually contain complex constructions which can not be easily processed through automation. From another point of view, the task of generating a set of simple sentences from a complex sentence can be considered as the task of generating a pool of possible sentences that a reader would assume to be true after reading the complex sentence.

In our project we use sentences in a body of a text to generate questions which are related to a topic of interest. The grammar of sentences in the body of the text may be complicated, or there may be some sentences with inner clauses. That is why sentences have to be simplified before we can generate more accurate questions. Why did not we do the same for basic questions? In the process of generating basic questions we only use the tagged name entities and we do nothing with the body of sentences themselves. The issue with sentence simplifiers is that when they receive a very complicated sentence and are unable to simplify it, they eliminate it, which can result in missing many named entities. Unlike basic questions, for generating specific questions we use other parts of the sentences, such as grammar, which is why it makes sense that simplifying them will help us to create more accurate questions.

To perfom the task of simplification we use a toolkit called Simplified Factual Statement Extraction (Heilman and Smith, 2010a). This model simplifies sentences by changing semantic and syntactic structures and eliminating phrase types. For example, if you give Sentence A to the simplifier, it will be changed to Sentence B and Sentence C.

Sentence A:

Acro dance is known by various other names including acrobatic dance and gymnastic dance, though it is most commonly referred to simply as acro by dancers and dance professionals.

Sentence B:

Acro dance is known by various other names including acrobatic dance and gymnastic dance.

Sentence C:

It is most commonly referred to simply as acro by dancers and dance professionals.

3.6 Semantic Role Tagger

The Natural Language Processing society has lately experienced a noticeable growth of concern in the area of semantic roles parsing (also known as semantic role labeling). The task of semantic role parsing is usually performed by recognizing all the predicates in a given sentence, and then, finding and classifying groups of words which represent the semantic roles (arguments) for each of the predicates. In other words, semantic role labeling can be considered as the process of allocating a WHO did WHAT to WHOM, WHY, WHERE, HOW, WHEN structure to a plain body of text. Semantic role parsing usually facilitates advancements for those algorithms which are designed to deal with Natural Language Processing tasks, such as question answering, information extraction, machine translation and summarization, by providing them with semantic structures.

Recently, a few attempts have been done to create hand-tagged corpora such as FrameNet (UC Berkeley) and PropBank (UPenn/Colorado) in order to encode such information. The reason to create those corpora is to make it more feasible for the Natural Language Processing community to train supervised machine learning classifiers which can be used to automatically tag a huge amount of unseen texts.

In our project we need to parse sentences in the text semantically. To do this, we use an Automatic Statistical SEmantic Role Tagger (ASSERT) [3]. ASSERT is known as an automatic statistical semantic role tagger, which is able to naturally annotate semantic arguments in a text. When a sentence is given to ASSERT, it applies a full syntactic analysis of that sentence, and identifies all of the verb predicates. Then, it extracts features for constituents within the parse tree relative to the predicate, and eventually identifies and tags the constituents with the appropriate semantic arguments. ASSERT has been trained to do the task of tagging:

- PropBank arguments
- Thematic roles
- Opinions, in plain text

For example if ASSERT is given an input sentence the output will be as shown below:

[3] Available at http://cemantix.org/software/assert.html.

Input:

Acro dance is known by various other names including acrobatic dance and gymnastic dance.

Output:

1) *[ARG1 Acro dance] is [TARGET known] [ARG0 by various other names including acrobatic dance and gymnastic dance]*

2) *Acro dance is known by [ARG2 various other names] [TARGET including] [ARG1 acrobatic dance and gymnastic dance]*

As we can see, the output of our semantic role tagger contains verbs (predicates) with their arguments (semantic roles).

3.7 Specific Question Generation

The second group of questions that our system generates and adds to the question pool is called specific questions. The main difference between specific questions and basic questions is that for specific questions it is possible to use the information inside the body of text to generate answers for these questions. In order to generate basic questions our system tags named entities and then uses them to fill the gaps in a group of already designed question templates so we cannot guarantee that there exists required information in the text to find the answers. In process of generating specific questions we replaced some tagged clauses with question words and consequently these clauses can be used to generate answers.

As we mentioned before, the output of the previous step is a set of simplified sentences which have been tagged by a semantic role tagger, therefore, their verbs and arguments can be used to generate specific questions. For instance, the output of the semantic role tagger system for the given sentence *Apple's first logo is designed by Jobs and Wayne.* is: *[ARG1 Apple 's first logo] is [TARGET designed] [ARG0 by Jobs and Wayne]* . As you can see the output contains one predicate (verb) with its semantic roles (arguments).

These semantic roles are used to create specific questions from the sentences. For example, we can exchange ARG1 with the question word *What* and generate a question as: *What is designed by Jobs and Wayne?* Similarly, ARG0 can be replaced and the question: *Who designed Apple's first logo?* can be generated. The semantic roles ARG0...ARG5 are named mandatory arguments. ASSERT has also some additional arguments or semantic role tags which are called optional arguments and start with the prefix ARGM. When we are provided with mandatory arguments, the choice of question word will depend on the named entity tag of the argument, such as *Who* for a person or *Where* for a location. In order to generate the specific questions, we apply 350 rules (written in Perl language) to transform the tagged sentences into questions (which can be found at Appendix D). Table 3.2 shows how we can replace different semantic roles by possible question words to generate a question.

Table 3.2: Semantic roles with possible question words

Arguments	**Question Words**
ARG0...ARG5	Who, Where, What, Which
ARGM-ADV	In what circumstances
ARGM-CAU	Why
ARGM-DIS	How
AGRM-EXT	To what extent
ARGM-LOC	Where
ARGM-MNR	How
ARGM-PNC	Why
ARGM-TMP	When

3.8 Summary

In this chapter we presented an overall view of the concepts of Topic to Question Generation task, and more detailed information about what steps have been taken to implement our project. Through different sections of this chapter efforts have been made to acquaint you with most of the toolkits that we have used, such as, the Name Entity Tagger, Sentence Simplifier and ASSERT.

We explained the reasons why we needed to use a NER tagger and the way that we benefited from a more advanced one which provided us with more tag-labels. Further, sentence simplification was one of the key points in our system to generate more accurate questions, therefore, we introduced a well-known simplifier which breaks a complex sentence to a set of simple well-formed sentences. To generate specific questions ASSERT was the tagger that we used to tag the simplified sentences semantically, though it can be used to perform other tasks too.

As we mentioned earlier, our proposed Automatically Topic to Question Generation system consists of six steps:

- Name Entity Tagging
- Basic Question Generation
- Sentence Simplification
- Predicates & Arguments Tagging
- Specific Question Generation
- Ranking the Generated Questions

The next chapter will concentrate on the last step of our system which deals with ranking the generated questions. The considered criteria of how to rank the questions will be fully explained through the next chapter.

Chapter 4

Ranking Generated Questions

4.1 Introduction

As we mentioned through previous chapters, one of the key points for search engines is to receive well-formed queries in order to provide users with their required data. Well-formed queries can be considered as ones that have explanatory information, which will clarify the users' needs when using search engines. Our goal is to generate some well-formed queries and offer them to the users in order to help them to gain needed information. However, after generating questions we do need to rank them. Ranking should be done because the number of generated questions is usually too large to effectively show all of them to the user, so we must rank and show the top N ones (N is the required number of questions to be shown). Here is an example showing a topic of interest and its related body of text, and some generated questions for this topic (more of which are shown in Appendix C):

Topic of Interest: *Traffic light*

Related Body of Text:

On 10 December 1868, the first traffic lights were installed outside the British Houses of Parliament in London, by the railway engineer J. P. Knight. They resembled railway signals of the time, with semaphore arms and red and green gas lamps for night use. The gas lantern was turned with a lever at its base so that the appropriate light faced traffic. Unfortunately, it exploded on 2 January 1869, injuring the policeman who was operating it. The modern electric traffic light is an American invention. As early as 1912 in Salt Lake

City, Utah, policeman Lester Wire invented the first red-green electric traffic lights. On 5 August 1914, the American Traffic Signal Company installed a traffic signal system on the corner of 105th Street and Euclid Avenue in Cleveland, Ohio. It had two colors, red and green, and a buzzer, based on the design of James Hoge, to provide a warning for color changes. The design by James Hoge (USPTO # 1251666 Sept. 22, 1913) allowed Police and Fire stations to control the signals in case of emergency. The first four-way, three-color traffic light was created by police officer William Potts in Detroit in 1920. In 1923, Garrett Morgan patented a traffic signal device, although it was not a precursor of the modern traffic light. Ashville, Ohio claims to be the location of the oldest working traffic light in the United States, used at an intersection of public roads until 1982 when it was moved to a local museum. The first interconnected traffic signal system was installed in Salt Lake City in 1917, with six connected intersections controlled simultaneously from a manual switch. Automatic control of interconnected traffic lights was introduced March 1922 in Houston, Texas. The first automatic experimental traffic lights in England were deployed in Wolverhampton in 1927.

Number of Generated Questions: 771

Sample of Generated Questions:

- *What was turned with a lever at its base so that the appropriate light faced traffic ?*
- *When automatic control of interconnected traffic lights was introduced in Houston ?*
- *When the first automatic experimental traffic lights in England were deployed in Wolverhampton ?*
- *Where the first automatic experimental traffic lights in England were deployed in 1927 ?*
- *When the first traffic lights were installed outside the British Houses of Parliament in London ?*

- *Who claims to be the location of the oldest working traffic light in the United States until 1982 when it was moved to a local museum ?*
- *When Ohio claims to be the location of the oldest working traffic light in the United States until when it was moved to a local museum ?*
- *Who invented the first red-green electric traffic lights as early as 1912 in Salt Lake City policeman ?*
- *Where automatic control of interconnected traffic lights was introduced March 1922 ?*
- *When Lester Wire invented the first red-green electric traffic lights ?*
- *Who installed a traffic signal system on the corner of 105th Street and Euclid Avenue in Cleveland Ohio on 5 ?*
- *Where the first interconnected traffic signal system was installed in 1917 ?*
- *What is the history of American Traffic Signal Company organization ?*
- *Who patented a traffic signal device ?*

Given that the total number of generated questions is 771 we need to provide ranking, in order to select the most relevant of them.

4.2 Importance of Generated Questions

4.2.1 Introduction

Beside grammatical correctness we have another scale to rank the generated questions which is importance and commonality. Now the question is, what is the touchstone to figure out which one is more important and which one is not?

Let us go through an example. Imagine that we want to rank the questions for a finance company. It would make more sense if we give higher scores to the ones which are related to financial topics and advance them to the top of the question list. In our case, we decided to score the questions according to their popularity. In other words, our proposed algorithm predicts what the user might be looking for, by investigating other questions that are already asked by other people. Then, this knowledge can be used to rank the generated questions.

In previous research done by Hasan (2013) the importance of the generated questions was estimated by their similarities to the texts (from where they were generated). We believe that there is another source to use in order to do the task of ranking. Nowadays, it is becoming a common practice for people to ask their questions in online forums, which are called Community-Based Question Answering (CQA) systems, such as the Yahoo! Answers' web site (which will be thoroughly introduced at the next section). In such forums, we can easily find thousands of questions being asked and answered by people around the world. We believe that there are many common questions being asked by people. We can use this information to study what people need and what they are mostly curious about. The key point is that people using CQA systems know that their questions are going to be answered by human beings, so they use more informative sentences to generate them. That is why questions being asked at forums are mostly well-formed and explanatory enough. Now in our system, we determine if the CQA questions are similar to the search engine queries so we can extract additional information about the users' probable interests.

4.2.2 Yahoo! Answers

Introduction

One of the most important parts of our algorithm is to use a rich CQA system. Thit is why we decided to use the Yahoo! Answers dataset as one. Yahoo! Answers is growing quickly. It is suggested that researchers increasingly use its dataset and it is becoming a popular source of information, such as advices or opinions (Liu and Agichtein, 2008). Yahoo! Answers (previously known as Yahoo! Q & A) was started by Yahoo! in June-

28-2005. It is a community-driven knowledge market (question-and-answer) website. This website allows users to both respond to questions asked by people and also forward questions to be answered by other people. As long as questions being asked in Yahoo! Answers do not violate its community guidelines they are all allowed.

Users

The proficiency and seriousness of the Yahoo! Answers' users can play a major role in the efficiency of our system. Therefore, we accomplished a research about this website's users. Since finding a good answer for a question is one of top goals for this website, in order to encourage better answers, effective participants are occasionally featured on the Blog of Yahoo! Answers. Yahoo! Answers gives the chance of gaining points in order to encourage users' participation. A number of researchers have been looking at the structure of the Yahoo! Answers' community and the interaction between users who ask or respond to questions. According to the typology studies of users on this website, it has been revealed that some people answer from their personal knowledge (specialists), while other users benefit from some external sources to make answers (synthesists), and it is observed that synthesists tend to accumulate more reward points.

Type of Information

Yahoo! Answers is a web site where people are able to post their questions and answers, all of which are publicly available to anyone who wants to use or download them. The data that we use in our experiments is Yahoo! Answers corpus as of 10/25/2007. It contains 4,483,032 questions and their corresponding answers. Additionally the corpus includes metadata, such as category, sub-category and best answer. All user ids in this corpus were anonymized so that no identifying information is revealed and no personal information is included.

Structure of Questions

In Yahoo! Answers people usually ask their questions in two steps. First, they ask a short and informative question which is called the subject. Then, they may attempt to explain their questions in a few sentences, which is called the content. The content part does not often provide more information. As we discovered, because people have only 140 characters for the subject part and they know that their questions are going to be answered by other people they usually try to ask a short and informative question as a subject. In the content part, which is not even mandatory, they use more sentences mostly to show their feelings. We draw your attention to the question below which has been asked on Yahoo! Answers:

Subject: *How do I make my parents buy me a laptop?*

Content: *Ok so I used to have a laptop but it broke. AND I DIDNT EVEN BREAK IT and my parents were mad at me! And when I try asking them if I'm getting a laptop again their like you have to wait til next year, and I asked why and they said because I have to save my money by getting a job? I'm too young to get a job, I'm 13, I really want a laptop before Christmas because I usually make these DVDs on my laptop to watch on Christmas. Please help I really want a laptop again!!*

Other Languages

Information presented on this website is available in 12 languages, but there are several sites which operate different platforms that permit non-Latin characters. For example, in Japan there is a platform known as Yahoo! Chiebukuro.

4.2.3 Description of Importance Estimation Algorithm

As we explained in the previous chapters, our thesis is about a complete system which is supposed to be given an informative text related to a user's query. Then it will create a pool of well-formed and explanatory questions that can be used by the user as her/his query or as a guide about how to write a good query. Generally we can split our system into two phases:

1. Generating Questions
2. Ranking Questions

In Chapter 3 we went through the first phase and illustrated that how our system utilizes named entities and predicate arguments in sentences in order to generate basic and specific questions. At the beginning of current Chapter we explained the main idea about the algorithm that we created to estimate the importance of the generated questions. Now we will go through all of the steps of this algorithm and show how it benefits from the CQA database.

The goal of our proposed algorithm is to estimate the importance of a group of questions as one of the steps required to do the task of ranking. At this point, we have generated a pool of questions and need to rank them because the number of questions is usually too large to be able to show all of them to the users. As we discussed earlier, the algorithm utilizes questions being asked in Yahoo! Answers (which are called subjects) to figure out how common the generated questions are.

To begin, we extract all subjects from Yahoo! Answers database. Then, when a user performs a Search Engine Query (SEQ), we calculate the semantic similarity between the SEQ and each extracted subject. Then we store the top scored subjects in an array, named Top-Subjects. We also save the scores of these Top-Subjects in another array called Top-Subjects-Scores. At this step for the first generated question we find its similarity scores with all Top-Subjects. We store these scores in an array called Generated-Question-Similarities-to-Top-Subjects. Finally, to obtain an overall score we take the average of all scores in both vectors Top-Subjects-Scores & Generated-Question-Similarities-to-Top-Subjects. At this point, we have one score showing us how similar the Generated-Question is to the questions that people have asked in Yahoo! Answers. The same steps will be taken for all generated questions, resulting in one similarity score for each one. These similarity scores show the importance of each question.

Now as we mentioned earlier in Chapter 1, another criteria in ranking is the grammatical correctness of the generated questions. We also need to score the questions according to their syntactic correctness (which we will explain in Section 4.3). After calculating both groups of scores (Importance & Grammatical Correctness) we sort the Generated-Questions by these scores and show the user as many top questions as required. We draw your attention to the pseudocode below that show the importance estimation algorithm:

Algorithm 1 Importance Estimation Algorithm

```
1:  Input 1: Yahoo! Answers Corpus as of 10/25/2007
2:  subjects ← a list of all subjects from Yahoo! Answers dataset
3:  Input 2: User's Search Engine Query (SEQ)
4:  i = 1
5:  for <All Subjects> do
6:      Semantic-Similarity-Score Between $i_{th}$ Subject and SEQ
7:      i++.
8:  end for
9:  Top-Subjects = Top-Scored-Subjects
10: Top-Subjects-Scores = Scores-of-Top-Subjects
11: j = 1
12: for <Generated Questions> do
13:     i = 1
14:     for <Top Subjects> do
15:         Semantic-Similarity-Score Between $i_{th}$ Top-Subject and $j_{th}$ Generated-
16:           Question
17:         i++;
18:     end for
19:     Final-Score = Average(Generated-Question-Similarities-to-Top-Subjects & Top-
20:       Subjects-Scores)
21:     j++;
22: end for
```

4.2.4 Semantic Similarity

As mentioned in the previous section, we calculate semantic similarity to find the importance factor. Here, we give more information about semantic similarity in general. Semantic similarity (also known as semantic relatedness) can be considered as a metric which has been defined over a set of terms or documents. The main idea of semantic similarity is to measure distances between documents or terms based on their likeness of meaning. In other words, semantic similarity refers to the concept of ”How much does term M have to do with term N?”. This question is usually answered by a number between -1 and 1, or 0 and 1, where 1 means very high similarity.

The semantic similarity concept is being used in different areas of natural language processing. Natural Language Processing (NLP) is known as a field of computer science which has been related to human–computer interaction. As an example of semantic similarity usage imagine that we are computer users searching for some information about a specific topic. Now, if we find one information resource which we are interested in, it is often of immediate interest to discover other similar sources. Therefore, we can benefit from semantic similarities to find similar data by content.

4.3 Judging Syntactic Correctness

4.3.1 Introduction

It is possible that some of the generated questions are syntactically incorrect. This may happen because of the process of automatic question generation. It is strongly believed that a question has a similar syntactic structure to the sentences from where it is generated (Hasan, 2013). Therefore, to judge the syntactic correctness of each generated question, we apply tree kernel functions (Collins and Duffy, 2001) in order to compute the syntactic similarity between each question and its associated body of text (Hasan, 2013). To measure the syntactic similarity between two sentences we first parse them syntactically which results in a parse tree for each sentence. Then, we apply tree kernel functions to these trees. The tree kernel function calculates the syntactic similarity score between each sentence in the given body of text and the generated question. Each sentence contributes a score to each question. Then, the questions are ranked by considering the average of similarity scores.

4.3.2 Syntactic Similarity

Content analysis can be considered as an advanced area of study. One of the subjects in content analysis is to assess the syntactic similarity of a text against another one. A given body of text may contain hidden information such as syntactic structure. Calculating syntactic similarities of the sentences can be used for many purposes such as eliminating duplicate text or clustering sentences.

4.3.3 Parser

A syntactic parser is a program which extracts the grammatical structure of a given sentence. For instance, a parser may provide us with information such as which words could be the subject or object of a verb, or which sets of words are together as a phrase. Probabilistic parsers benefit from a set of language knowledge that is gained from hand-parsed sentences in order to construct the most likely analysis of a new sentence. The parsers development was one of the biggest breakthroughs in the field of Natural Language Processing in the 1990s. Although they still make some mistakes, they commonly work acceptable. In our project we used the Stanford parser to parse sentences in the given texts and generated questions to use them as inputs for the tree kernel function. Below find an example of a sentence and its parse tree.

Input:

Acro dance is a style of dance that combines classical dance technique with precision acrobatic elements.

Output:

```
.(ROOT
.   (S
.       (NP (NNP Acro)(NN dance))
.       (VP (VBZ is)
.             (NP
.                (NP (DT a)(NN style))
.                (PP (IN of)
.                   (NP (NN dance)))
.                (SBAR
.                   (WHNP (WDT that))
.                   (S
.                      (VP (VBZ combines)
.                         (NP (JJ classical)(NN dance)(NN technique))
.                         (PP (IN with)
.                            (NP (NN precision)(NN acrobatic)(NNS elements))))))))
.       (. .)))
```

4.4 Summary

As we mentioned in previous chapters, the web engines' users may face some difficulties regarding finding documents that they are looking for in online areas. It may happen because web engines are very sensitive to the queries and they need informative inputs in order to be able to extract the required information. Our goal is to help users to generate better and more explanatory queries, and consequently receive more relevant data. Our proposed system first generates a pool of questions which are closely related to the users' interests and can be used as their queries. At this point, the number of generated questions can be too large to show all of them to the users, so then our system sorts them according to their importance and grammatical correctness.

In this chapter we first explained our proposed ranking algorithm which was used to score a set of generated questions to find their importance. We used the Yahoo! Answers dataset as our source of commonly asked questions. We explained why we decided to use this dataset and what are its shortcomings and strengths. Further, we gave a step-by-step explanation about the algorithm by which we score the generated questions according to their importance. Then, we introduced the tool-kit that was used to perform the task of semantic similarity estimation. Afterwards, we explained that why we also needed to score the questions by their grammatical correctness, and then introduced the tool-kit used to parse sentences. As we found it essential to make a better understanding of our project we described concepts of semantic and syntactic similarity and also the tree kernel function.

Chapter 5

Experiments and Evaluation

5.1 Introduction

In this chapter we introduce the methodology that we used in order to evaluate our system. We also explain how we benefited from other question generation systems and the corpus from where we generated the questions. At the end we will provide and discuss the result of the comparison and also discuss some key points about how to advance our system as a future job.

5.2 Corpus

In our experiments, we use the dataset from the Question Generation Shared Task and Evaluation Challenge (Rus et al., 2010) to tackle the task of automatically generating questions. The dataset consists of 60 paragraphs, each related to 60 topics (which can be found in Appendix C). They are originally selected from several sources such as OpenLearn, Wikipedia and Yahoo! Answers. The paragraphs are constructed from approximately 57 sentences, for a total number of 100200 tokens including punctuations.

As mentioned in prior chapters, to do our experiments we assume that there exists a text related to each query containing useful information about it. Therefore, we consider the topics as queries and treat paragraphs as the associated body of the texts. There is an example of a topic and its related text below:

Apple Inc. logos

Apple's first logo, designed by Jobs and Wayne, depicts Sir Isaac Newton sitting under an apple tree. Almost immediately, though, this was replaced by Rob Janoff's "rainbow Apple", the now-familiar rainbow-colored silhouette of an apple with a bite taken out of it. Janoff presented Jobs with several different monochromatic themes for the "bitten" logo, and Jobs immediately took a liking to it. While Jobs liked the logo, he insisted it be in color to humanize the company. The Apple logo was designed with a bite so that it would be recognized as an apple rather than a cherry. The colored stripes were conceived to make the logo more accessible, and to represent the fact the monitor could reproduce images in color. In 1998, with the roll-out of the new iMac, Apple discontinued the rainbow theme and began to use monochromatic themes, nearly identical in shape to its previous rainbow incarnation.

5.3 Evaluation Setup

5.3.1 Methodology

Our methodology to evaluate the performance of our automated question generation system is inspired by Hasan (2013). Three unknown native English speakers were chosen to judge the result of our system. They were asked to score the generated questions according to two criteria:

- **Syntactic Correctness**
- **Topic Relevance**

The judges give scores between 1 (very poor) and 5 (very good). There are four scores for each generated question. To evaluate the topic relevance criterion, the judges were given three aspects, and they scored each question according to each aspect. The aspects are:

1. **Questions' semantic correctness:** The judges will score questions considering if they make sense, for example a question like: *"Is Lethbridge weather a good person?"* will get a lower score.

2. **Question type correctness:** In here the question words are under consideration, for example, the score of *"How is Lethbridge weather?"* will get a lower score rather than *"How cold/hot/sunny is lethbridge weather?"*

3. **Clarity of referential:** As the questions are generated automatically, sometimes it happens that they are too long. Users may lose their tracks while reading the questions, so in here the judges score them if the points that the questions are referring are clear.

For syntactic correctness, they scored the generated questions by considering if they are grammatically correct or not. Then, the average of the judges' scores is calculated for each question.

5.3.2 Systems for Comparison

To evaluate our system we compare it with the state-of-the-art QG system proposed by Hasan (2013). To do so, we use a publicly available QG system published by Heilman and Smith (2010b) as a benchmark. Hasan (2013) generated a total of 2186 questions from 50 randomly chosen texts, then ranked and evaluated the top-ranked 20% of questions. In our evaluation system we are generating the questions from 10 randomly chosen texts and then selecting the top-ten ranked questions. We also generate questions for the same texts using the Heilman QG toolkit, and again select the top-ten ranked ones. The judges were presented with 20 questions per text. The top-ten from our system and the top-ten from Heilman's system. We have ten texts, so the total number of questions is 200 for each judge. After comparing our system with Heilman, we calculate our system advancement in comparison with the one created by Hasan (2013).

5.3.3 Results and Discussion

The main goal of our proposed system was to generate questions, which are related to a given topic. To do this, we generated a pool of basic and specific questions. To generate the questions, we used named entities and predicates arguments, which were in the given body of the text. As the number of generated questions might be large, we ranked them by considering similar questions, which were asked in a CQA system and also their grammatical correctness. Table 5.1 lists the average of syntactic correctness and topic relevance scores for each system. These results confirm that our proposed automated question generation system outperforms the Heilman's system by 29.39%, and 18.71%, and over the Hasan's system by 25.38%, and 14.04%, respectively.

Table 5.1: Syntactic correctness and topic relevance scores

Systems	Syntactic Correctness	Topic Relevance
Heilman and Smith	3.13	3.42
State-of-the-art	3.23	3.56
Proposed Question Generation System	4.05	4.06

5.4 Summary

Through the chapters of this thesis we explained all the reasons that encouraged us to design and implement a system from which online researchers can use to generate more informative and explanatory queries. We walked through all the steps of our system from tagging a set of information to generating a pool of well-formed questions, and finally the ranking step where they are scored according to some criteria. We also provided the information about all of the toolkits and databases that we used in different parts of our algorithm. In the current Chapter, the methodology which we used to evaluate our system was introduced and at the end the results of the evaluation were discussed.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

It is undeniable that in our century asking questions and seeking for needed data is one the most essential activities that people need to perform in order to accomplish their lives' tasks. For example, in educational systems, imagine how much students will benefit from online areas to ask their questions and look for answers which will fit their needs. However, the most considerable issue is the amount of time that online researchers need to spend in order to receive the information that they are looking for. This problem would not happen if people could directly ask other people who are already aware of the answers. However, human resources are not accessible all the time and they usually can not provide us with a very comprehensive data as they may lack it or do not have enough time to transfer all the data in a timely manner. As a solution, we believe that if we use online resources in a more efficient way they could be much less time consuming and at the same time provide users with adequate amount of information and enough time to gain those required information. This task can be easily done if we know how to use online facilities. For example, if you want to ask your questions in an online forum first you should know how to ask them so that no one will be offended and also understand the questions so that they can answer them. Beside forums, search engines are another set of online researching toolkits that are receiving a huge amount of attention from users. The same as forums, search engines need to receive an informative query in order to be able to understand your needs and extract the information that most fits your question. Now the question is how to compose a well-formed query that search engines can realize our needs. To accomplish this task we created a complete system which will be provided with a query from users and then show them a set of questions that they can use to write a more explanatory query and express themselves better. Our assumption is that there already exists a document related to the query.

Through the chapters of this thesis we fully explain all details and steps taken to create this system. We also provided some basic information in the Natural Language Processing area that was crucial in order to comprehend the logic behind our proposed question generation toolkit.

6.2 Future Work

As we explained in this thesis, different toolkits have been used to perform each step of our task. We strongly believe that any advancement in any of these steps can noticeably improve the output of our system. For example, at the step that we want to generate basic questions we need to tag named entities. A key point at this time is the accuracy of the tagged entities beside the amount of different labels that the toolkit provides us. Imagine that in a sentence we have the word ”Washington”. As you can see this word can be used as either a person’s name or a place. Therefore in our basic question generation part we would be much more accurate if we knew which meaning of this word is meant in the sentence.

Another future work could be about the database that we used. We discussed it earlier that in order to do the task of ranking we use the Yahoo! Answer database as our source of already asked questions. At this step, the amount of CQA questions that we are considering to figure out how important our generated questions are could have a good influence on the ranking algorithm. So one way to advance our system could be to monitor the effect of the amount of input data for the ranking section.

As we mentioned in chapter 4, the Yahoo! Answers' database also provides us with some other information related to the questions asked in that website. For example, it is said that the asked questions are under different categories or sub-categories in the Yahoo! Answers. We believe that this data can be used to perform the task of ranking generated questions if the search engine is given the user's interest category. In other words, if we knew that someone is asking a question which is under a category called x we could try to use CQA questions in that category to estimate how important the generated questions are.

Another area that may advance our system's output is sentence simplification. As we discussed earlier, one of the shortcomings of the present toolkits to simplify sentences is that they usually eliminate sentences thar are found to be too complicated to be simplified. Consequently, the questions which could be generated from them might be missing from the question pool.

Bibliography

M. Agarwal and P. Mannem. Automatic gap-fill question generation from text books. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 56–64. Association for Computational Linguistics, 2011.

I. Aldabe, M. L. de Lacalle, M. Maritxalar, E. Martinez, and L. Uria. Arikiturri: an automatic question generator based on corpora and nlp techniques. In *Intelligent Tutoring Systems*, pages 584–594. Springer, 2006.

H. Ali, Y. Chali, and S. A. Hasan. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 58–67, 2010.

A. Andrenucci and E. Sneiders. Automated question answering: Review of the main approaches. In *ICITA (1)*, pages 514–519, 2005.

L. Becker, S. Basu, and L. Vanderwende. Mind the gap: learning to choose gaps for question generation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 742–751. Association for Computational Linguistics, 2012.

L. Bednarik and L. Kovacs. Implementation and assessment of the automatic question generation module. In *Cognitive Infocommunications (CogInfoCom), 2012 IEEE 3rd International Conference on*, pages 687–690. IEEE, 2012.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

K. E. Boyer and P. Piwek. Proceedings of qg2010: The third workshop on question generation. 2010.

J. C. Brown, G. A. Frishkoff, and M. Eskenazi. Automatic question generation for vocabulary assessment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 819–826. Association for Computational Linguistics, 2005.

C.-Y. Chen, H.-C. Liou, and J. S. Chang. Fast: an automatic generation system for grammar tests. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 1–4. Association for Computational Linguistics, 2006.

W. Chen. Aist g, mostow j (2009) generating questions automatically from informational text. In *Proceedings of the 2nd Workshop on Question Generation*, pages 17–24, 2009.

M. Collins and N. Duffy. Convolution kernels for natural language. In *Advances in neural information processing systems*, pages 625–632, 2001.

D. M. Gates. Automatically generating reading comprehension look-back strategy: Questions from expository texts. Technical report, DTIC Document, 2008.

S. S. A. Hasan. *Complex question answering: minimizing the gaps and beyond.* PhD thesis, Lethbridge, Alta.: University of Lethbridge, Dept. of Mathematics and Computer Science, 2013.

M. Heilman and N. A. Smith. Extracting simplified statements for factual question generation. In *Proceedings of QG2010: The Third Workshop on Ques-tion Generation*, page 11, 2010a.

M. Heilman and N. A. Smith. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617. Association for Computational Linguistics, 2010b.

S. Kalady, A. Elikkottil, and R. Das. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 1–10. questiongeneration. org, 2010.

I. Labutov, S. Basu, and L. Vanderwende. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 889–898. Association for Computational Linguistics, 2015.

T. K. Landauer. Latent semantic analysis. *Encyclopedia of Cognitive Science*, 2006.

C.-Y. Lin. Automatic question generation from queries. In *Workshop on the Question Generation Shared Task*, pages 156–164, 2008.

Y.-C. Lin, L.-C. Sung, and M. C. Chen. An automatic multiple-choice question generation scheme for english adjective understanding. In *Workshop on Modeling, Management and Generation of Problems/Questions in eLearning, the 15th International Conference on Computers in Education (ICCE 2007)*, pages 137–142, 2007.

D. Lindberg, F. Popowich, J. Nesbit, and P. Winne. Generating natural language questions to support learning on-line. 2013.

M. Liu, R. A. Calvo, and V. Rus. Automatic question generation for literature review writing support. In *Intelligent Tutoring Systems*, pages 45–54. Springer, 2010.

Y. Liu and E. Agichtein. On the evolution of the yahoo! answers qa community. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 737–738. ACM, 2008.

P. Mannem, R. Prasad, and A. Joshi. Question generation from paragraphs at upenn: Qg-stec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 84–91, 2010.

K. Mazidi and R. D. Nielsen. Linguistic considerations in automatic question generation. pages 321–326, 2014.

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to wordnet: An on-line lexical database*. *International journal of lexicography*, 3(4):235–244, 1990.

R. Mitkov and L. A. Ha. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing-Volume 2*, pages 17–22. Association for Computational Linguistics, 2003.

N. Myller. Automatic generation of prediction questions during program visualization. *Electronic Notes in Theoretical Computer Science*, 178:43–49, 2007.

S. Ou, C. Orasan, D. Mekhaldi, and L. Hasler. Automatic question pattern generation for ontology-based question answering. In *FLAIRS Conference*, pages 183–188, 2008.

P. Piwek and S. Stoyanchev. Question generation in the coda project. 2010.

E. Reiter, C. Mellish, and J. Levine. Automatic generation of on-line documentation in the idas project. In *Proceedings of the third conference on Applied natural language processing*, pages 64–71. Association for Computational Linguistics, 1992.

O. Rokhlenko and I. Szpektor. Generating synthetic comparable questions for news articles. In *ACL (1)*, pages 742–751, 2013.

V. Rus and C. G. Arthur. The question generation shared task and evaluation challenge. In *The University of Memphis. National Science Foundation*. Citeseer, 2009.

V. Rus, B. Wyse, P. Piwek, M. Lintean, S. Stoyanchev, and C. Moldovan. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 251–257. Association for Computational Linguistics, 2010.

V. Rus, M. C. Lintean, R. Banjade, N. B. Niraula, and D. Stefanescu. Semilar: The semantic similarity toolkit. In *ACL (Conference System Demonstrations)*, pages 163–168. Citeseer, 2013a.

V. Rus, N. Niraula, and R. Banjade. Similarity measures based on latent dirichlet allocation. In *Computational Linguistics and Intelligent Text Processing*, pages 459–470. Springer, 2013b.

R. Soricut and E. Brill. Automatic question answering: Beyond the factoid. In *HLT-NAACL*, pages 57–64, 2004.

R. Soricut and E. Brill. Automatic question answering using the web: Beyond the factoid. *Information Retrieval*, 9(2):191–206, 2006.

W. Wang, T. Hao, and W. Liu. Automatic question generation for learning evaluation in medicine. In *Advances in Web Based Learning–ICWL 2007*, pages 242–251. Springer, 2008.

Y. Xu, A. Goldie, and S. Seneff. Automatic question generation and answer judging: a q&a game for language learning. In *SLaTE*, pages 57–60, 2009.

Appendix A

Basic Question Templates

Our goal is to generate a question pool and then rank them and show the top-ranked ones to search engines' users. First type of questions are called basic questions, for which we may not have the answers, in other words, the body of texts (where the basic questions have been generated from) may not have the needed information to generate the required answers. The second type of questions are called specific questions, unlike basic questions it is possible to use the body of texts to exploit adequate amount of information and then use them to generated the answers. To generate our basic questions we first extracted all named entities in the topics of interests and their related body of texts. Then according to each label tag (such as, EVENT, GPE, Language, etc.) we created a set of templates having a gap inside which can be filled by an appropriate tagged named entity. In here we expose the templates that we used, we draw your attention to the ”...” in each template which shows the gap. At this table you can find the tag labels on the left column and their templates on the right one.

Table A.1: The question templates used to generate basic questions

Tag Labels	Basic Question Templates
EVENT	Is ... over?
	How much money is spent for ... ?
	What do we know about ... ?
	How many countries are involved in ... ?
	What are the most important consequences of ... ?
	What is the history of ... ?
	Are specific types/races of people involved in ... ?
	Are there any specific organisations or companies involved in ... ?
	Do we have any similar events to ... ?
	Are there any famous people involed in ... ?
	Have there ever been any protests against ... ?
	Has the name of ... ever changed ?
	Does ... have any other names ?
	Where is ... started?
	Where has ... taken place?
	Where is ... finished?
	Who have participated in ... ?
	Who have organized ... ?
	Who first named ... ?
	For whom was ... advantageous or disadvantageous ?
	When is ... started ?
	When is ... finished ?
	What happens in ... ?
	what major events happened at ... ?
	What are the reasons of ... event?
	What type of facilities have been used in ... ?
	What is the best source to gain more information about ... ?
	Which political or religious communities have participated in ... ?
	Why has ... started ?
	Why do we have to know about ... ?
	Why is it named ... ?
	How long is ... ?
	How old is ... ?
	How someone could take part in ... ?
	How many people have been involved in ... ?

Continued on next page

Table A.1 – continued from previous page

Tag Labels	Basic Question Template
Facility	Have there ever been any protests or strike against ... ? Why is ... named so ? Who first named ... ? Has the name of ... ever changed ? How big is ... ? Are there any similar Structures to ... ? Has anybody injured in process of building ... ? How much money has been spent to build ... ? How much money is spent yearly to maintain ... ? How did they build ... ? Does ... have any other names ? Where is ... located ? Who designed ... ? Who participated in building ... ? Who either works or lives in ... ? When did they start building ...? When did they finish building ...? What type of materials are used to build ...? What more should we know about ...? What were the reasons that ... was build ? what is the best source to gain more information about ... ? Why is ... build? How can someone get to ...? How old is ... ? How many people use ...? How long did it take to build ... ? How well-built is ... in comparison with others ?

Continued on next page

Table A.1 – continued from previous page

Tag Labels	Basic Question Template
GPE	Have there ever been any famous people living in ... ? What are the main industrial activities in ... ? Are there any underground reserves in ... ? Have ever been any specific diseases from ... ? What is the income levels in ... ? What are laws and regulations of ... ? What political parties are there in ... ? What religious group are there in ... ? How many languages are there in ... ? What is the educational level in ... ? What is the best souvenir from ... ? What is the traditional costume in ... ? Have there ever been any wars in ... ? Have there ever been any protests or strike in ... ? Have there ever been any natural disasters in ... ? What is the history of ... ? What type of educational system does ... have ? Is ... a country or a state/province or a city ? Are there any touristic places in ... ? Does ... have any other names ? where is ...? Where is ... bordered with ? Who named ... ? When is the best time to travel around ... ? What type/races of people live in ... ? What languages ...'s inhabitant speak ? What is climate like in ... ? What is the best way to travel around ... ? What type of wildlife exists in ... ? what is the best source to gain more information about ... ? Why is it named ... ? How many people live in ... ? How old is ... ?

Continued on next page

Table A.1 – continued from previous page

Tag Labels	Basic Question Template
Language	How many letters does ... language have ? Has ... language ever changed ? How difficult is learning ... language in comparison with others ? Are there any famous literary works in ... language ? What is the history of ... language ? Are there any similar languages to ... ? Does ... language have any other names ? Where is ... language spoken ? Where is the best place to learn ... ? Who spoke ... language ? Who created ... language ? What is the best way to learn ... ? What race of people speak ... ? What is the best source to gain more information about ... language? What is the category of ... language? Why is this language named ... ? How many people speak ... ? How old is ... language ?
Law	Does the law of ... have specific period ? Is the law of ... publicly published ? Have there ever been any protests or strike against the law of ... ? Has the law of ... ever changed ? Who are beneficiary of the law of ... ? Who are disadvantaged by the law of ... ? Who named the law of ... ? Does ... have any other names ? Where was the law of ... passed ? When was the law of ... passed ? When was the law of ... executed ? Who are authorized to execute the law of ... ? Who are involved in the law of ... ? What were the reasons of having the law of ... ? What is the best source to gain more information about the law of ... ? What is the law of ... about? Why is it named ... ?

Continued on next page

Table A.1 – continued from previous page

Tag Labels	Basic Question Template
Location	Are there any touristic places in ... ? Is ... either mountain ranges or bodies of water ? Have ever been any wars for ... ? Have there ever been any natural disasters in ... ? Which country(ies) does ... belong to ? When was ... discovered ? Does ... have any other names ? Where is ... ? When is the best time to travel around ... ? Who first discovered ... ? Who named ... ? What type of wildlife exists in ... ? What is climate like in ... ? What is the best source to gain more information about ... ? Why is it named ... ? How big is ... ?
Money	How to transfer ... ? How to earn ... ? whose currency is ... ? What is the best source to gain more information about ... ? Why is it named ... ?
NORP nationalities religious political-groups	Are there any other names for ... people ? Have ever been any wars related to ... people ? Have ever been any strikes or protests having ... people involved ? What is the culture of ... people ? Do ... people have any leaders ? What are the main occupations of ... people ? Where do ... people live ? Who are named ... people? What language ... people speak ? What is the best source to gain more information about ... people ? What is ... people history ? How many people are ... ?

Continued on next page

Table A.1 – continued from previous page

Tag Labels	Basic Question Template
Ordinal	What is an ordinal number ? Why are ordinal numbers like ” ... ” written in this way ? What is the ordinal number ” ... ” in other languages? Is ” ... ” an ordinal number? What is the oridinal number after ... ? What is the ordinal number beffor ... ? Who named numbers like ” ... ” as ordinal numbers ? What is the best source to gain more information about ordinal numbers like ... ? Why are numbers like ” ... ” named ordinal ? How to use ordinal numbers like ” ... ” ?
Organization	Have ever been any strikes or protests against ... ? Is ... an organization ? Is ... sill working ? where is ... located ? Who work at ... ? Who are involved in ... ? Who founded ... ? When is ... founded ? What do they do at ... ? What were the reasons of founding ... ? What is the best source to gain more information about ... ? What is the best way to get to the ... ? How many people work at ... ? How old is ... ?
Person	Is ... a real or a fictional person ? Is ... still alived ? Where was ... born ? Where is ... from? Who knows ... ? Who is ... ? Who are family members of ... ? When was ... born ? What about the marital state of ... ? What is the best source to gain more information about ... ? What about occupation of ... ? What about ...’s sexuality ? What about ... age ?

Continued on next page

Table A.1 – continued from previous page

Tag Labels	**Basic Question Template**
Person	Where has ... been produced ? Where has ... been used ? Where can we buy ... ? Who produced ... ? Who have been useing ... ? When was ... produced first? What is ... made of ? What were the reasons of producing ... ? What is the best source to gain more information about ... ? Why someone may need ... ? How many people have been using ? How big is ... ? How much is ... ?
Quantity	What are the scales like " ... " used for ? What can be used to measure ... ? How to convert ... to other scales ? How to measure ... ?
Work Of Art	Is ... a work of art? Where was ... created ? Who created ... ? When was ... created ? What is the best source to gain more information about ... ? Why is ... created ? How old is ... ?

Appendix B

Samples of Generated Questions

In chapter 4 we showed an example of a topic of interest, its related body of text and some sample generated questions. In here you can find three examples about topics, texts and generated questions. As we mentioned before, we added basic questions to have more variety in our question pool. In the section below, in order to distinguish them, we put a star sign beside basic questions. As you can see adding them had a significant influence in our results.

Topic of Interest: *RELIGIOUS CRUSADES*
Related Body of Text:
The immediate cause of the First Crusade was the Byzantine emperor Alexius I's appeal to Pope Urban II for mercenaries to help him resist Muslim advances into territory of the Byzantine Empire. In 1071, at the Battle of Manzikert, the Byzantine Empire was defeated, which led to the loss of all of Asia Minor (modern Turkey) save the coastlands. Although attempts at reconciliation after the East-West Schism between the Catholic Western Church and the Eastern Orthodox Church had failed, Alexius I hoped for a positive response from Urban II and got it, although it turned out to be more expansive and less helpful than he had expected. The long history of losing territories to a religious enemy created in the Vatican a powerful motive to respond to emperor Alexius I's call for holy war to defend Christendom, and to recapture the lost lands starting with Jerusalem.
Number of Generated Questions: 462
Sample of Generated Questions:

- *What religious group are there in Jerusalem?
- *Are there any religious groups involved in the First Crusade organization?
- *What religious group are there in Christendom?
- *Are there any religious group that Alexius takes part in?
- When The Byzantine Empire was defeated at the Battle of Manzikert?
- *What do we know about the Battle of Manzikert?
- *What happens in the Battle of Manzikert?

- What is the traditional costume in the Byzantine Empire?
- *Where is the Battle of Manzikert started?
- What are laws and regulations of the Byzantine Empire?
- *Why do we have to know about Alexius?
- *Where has the Battle of Manzikert taken place?
- *Does the Battle of Manzikert have any other names?
- *Who are the leaders of the Eastern Orthodox Church organization?

Topic of Interest: *Silk*
Related Body of Text:
Silk is a natural protein fiber, some forms of which can be woven into textiles. The best-known type of silk is obtained from cocoons made by the larvae of the mulberry silkworm Bombyx mori reared in captivity (sericulture). The shimmering appearance of silk is due to the triangular prism-like structure of the silk fiber which allows silk cloth to refract incoming light at different angles thus producing different colors. Silks are produced by several other insects, but only the silk of moth caterpillars has been used for textile manufacture. There has been some research into other silks, which differ at the molecular level. Silks are mainly produced by the larvae of insects that complete metamorphosis, but also by some adult insects such as webspinners. Silk production is especially common in the Hymenoptera (bees, wasps, and ants), and is sometimes used in nest construction. Other types of arthropod produce silk, most notably various arachnids such as spiders.
Number of Generated Questions: 17
Sample of Generated Questions:

- When silk production is used in nest construction?
- *Do we have any information about what only the silk of moth caterpillars has been used?
- *What do we know about people who are involved in the fact that silks are produced?
- Where silk production is sometimes used?
- *Do we have any information about what the best-known type of silk is obtained?
- Who produce silk most notably various arachnids such as spiders?
- What are produced by several other insects?
- *Do we have any information about what the natural protein fiber is some forms of which can be woven?
- What has been used for textile manufacture?

- Where other silks differ?
- What could be used for textile manufacture?

Topic of Interest: *Wu Zetian*
Related Body of Text:
Wu Zetian entered the Tang palace at 13 and became a concubine of Emperor Taizong. She did not become a favorite of Taizong's, and after his death in 649, she might have been expected to spend the rest of her life as a Buddhist nun, like his other childless concubines. However, through an unlikely fortuity - Empress Wang, the wife and empress of Emperor Taizong's son and successor Emperor Gaozong, wanted another beautiful concubine to divert Emperor Gaozong's favors from Consort Xiao, with whom Empress Wang was having a desperate struggle - Wang had her brought back to the palace and made a concubine of Emperor Gaozong. Consort Wu proceeded to defeat both Empress Wang and Consort Xiao in the struggle for Emperor Gaozong's affection, and subsequently, both Empress Wang and Consort Xiao were killed, and she was made empress.
Number of Generated Questions: 172
Sample of Generated Questions:

- *What about the marital state of Wu Zetian?
- *What about occupation of Consort Wu?
- *What do we know about Wu Zetian?
- *Who is Wu Zetian?
- *Who is Consort Wu?
- *Have ever been any strikes or protests against Wu Zetian?
- Who proceeded to defeat both Empress Wang and Consort Xiao in the struggle for emperor Gaozong 's affection?
- *Do we have any information about what consort Wu proceeded?
- What is the best source to gain more information about Wu Zetian?
- *Is Wu Zetian a real or a fictional person?
- *Are there any political parties that Consort Wu takes part in?
- *Who are family members of Gaozong?
- *What is the best source to gain more information about Consort Xiao?
- *What do we know about Gaozong?

Appendix C

Topics of Interests and Related Body of Texts

Topic of Interest:
Silk
Related Body of Text:
Silk is a natural protein fiber, some forms of which can be woven into textiles. The best-known type of silk is obtained from cocoons made by the larvae of the mulberry silkworm Bombyx mori reared in captivity (sericulture). The shimmering appearance of silk is due to the triangular prism-like structure of the silk fiber which allows silk cloth to refract incoming light at different angles thus producing different colors. Silks are produced by several other insects, but only the silk of moth caterpillars has been used for textile manufacture. There has been some research into other silks, which differ at the molecular level. Silks are mainly produced by the larvae of insects that complete metamorphosis, but also by some adult insects such as webspinners. Silk production is especially common in the Hymenoptera (bees, wasps, and ants), and is sometimes used in nest construction. Other types of arthropod produce silk, most notably various arachnids such as spiders.

Topic of Interest:
Sextant
Related Body of Text:
A sextant is an instrument used to measure the angle between any two visible objects. Its primary use is to determine the angle between a celestial object and the horizon which is known as the altitude. Making this measurement is known as sighting the object, shooting the object, or taking a sight. The angle, and the time when it was measured, can be used to calculate a position line on a nautical or aeronautical chart. A common use of the sextant is to sight the sun at noon to find one's latitude. Since the sextant can be used to measure the angle between any two objects, it can be held horizontally to measure the angle between any two landmarks which will allow for calculation of a position on a chart. The scale of a sextant has a length of 1/6 of a full circle (60C); hence the sextant's name (sextans, -antis is the Latin word for "one sixth", "......." in Greek).

Topic of Interest:
Diamond
Related Body of Text:
Most natural diamonds are formed at high-pressure high-temperature conditions existing at depths of 140 to 190 kilometers (87 to 120 mi) in the Earth mantle. Carbon-containing minerals provide the carbon source, and the growth occurs over periods from 1 billion to 3.3 billion years (25% to 75% of the age of the Earth). Diamonds are brought close to the Earth surface through deep volcanic eruptions by a magma, which cools into igneous rocks known as kimberlites and lamproites. Diamonds can also be produced synthetically in a high-pressure high-temperature process which approximately simulates the conditions in the Earth mantle. An alternative, and completely different growth technique is chemical vapor deposition. Several non-diamond materials, which include cubic zirconia and silicon carbide and are often called diamond simulants, resemble diamond in appearance and many properties. Special gemological techniques have been specially developed to distinguish natural and synthetic diamonds and diamond simulants. Because of its extremely rigid lattice, diamonds can be contaminated by very few types of impurities, such as boron and nitrogen. Small amounts of defects or impurities (about one per million of lattice atoms) color diamond blue (boron), yellow (nitrogen), brown (lattice defects), green, purple, pink, orange or red.

Topic of Interest:
Photosynthesis
Related Body of Text:
Photosynthesis (from the Greek ....- [photo-], "light," and .... [synthesis], "putting together", "composition") is a process that converts carbon dioxide into organic compounds, especially sugars, using the energy from sunlight.[1] Photosynthesis is vital for life on Earth. As well as maintaining the normal level of oxygen in the atmosphere, nearly all life either depends on it directly as a source of energy, or indirectly as the ultimate source of the energy in their food (the exceptions are chemoautotrophs that live in rocks or around deep sea hydrothermal vents). The amount of energy trapped by photosynthesis is immense, approximately 100 terawatts: which is about six times larger than the power consumption of human civilization. As well as energy, photosynthesis is also the source of the carbon in all the organic compounds within organisms' bodies. In all, photosynthetic organisms convert around 100,000,000,000 tonnes of carbon into biomass per year

Topic of Interest:
GPS
Related Body of Text:
A GPS receiver calculates its position by precisely timing the signals sent by the GPS satellites high above the Earth. Each satellite continually transmits messages which include: the

time the message was transmitted, precise orbital information, the general system health and rough orbits of all GPS satellites (the almanac). The receiver utilizes the messages it receives to determine the transit time of each message and computes the distances to each satellite. These distances along with the satellites' locations are used with the possible aid of trilateration to compute the position of the receiver. This position is then displayed, perhaps with a moving map display or latitude and longitude; elevation information may be included. The GPS signal allows to repeat this calculation every 6 seconds. Many GPS units show derived information such as direction and speed, calculated from position changes. Three satellites might seem enough to solve for position, since space has three dimensions and a position on the Earth's surface can be assumed. However, even a very small clock error multiplied by the very large speed of light-the speed at which satellite signals propagate-results in a large positional error. Therefore receivers use four or more satellites to solve for the receiver's location and time.

Topic of Interest:
Lens optics Types of Lenses
Related Body of Text:
Lenses are classified by the curvature of the two optical surfaces. A lens is biconvex (or double convex, or just convex) if both surfaces are convex. If both surfaces have the same radius of curvature, the lens is equiconvex. A lens with two concave surfaces is biconcave (or just concave). If one of the surfaces is flat, the lens is plano-convex or plano-concave depending on the curvature of the other surface. A lens with one convex and one concave side is convex-concave or meniscus. It is this type of lens that is most commonly used in corrective lenses. If the lens is biconvex or plano-convex, a collimated or parallel beam of light travelling parallel to the lens axis and passing through the lens will be converged (or focused) to a spot on the axis, at a certain distance behind the lens (known as the focal length). In this case, the lens is called a positive or converging lens. If the lens is biconcave or plano-concave, a collimated beam of light passing through the lens is diverged (spread); the lens is thus called a negative or diverging lens. The beam after passing through the lens appears to be emanating from a particular point on the axis in front of the lens; the distance from this point to the lens is also known as the focal length, although it is negative with respect to the focal length of a converging lens.

Topic of Interest:
Enzyme Biological function
Related Body of Text:
Enzymes are mainly proteins, that catalyze (i.e., increase the rates of) chemical reactions. An important function of enzymes is in the digestive systems of animals. Enzymes such as amylases and proteases break down large molecules (starch or proteins, respectively) into smaller ones, so they can be absorbed by the intestines. Starch molecules, for example, are too large to be absorbed from the intestine, but enzymes hydrolyse the starch chains into

smaller molecules such as maltose and eventually glucose, which can then be absorbed. Different enzymes digest different food substances. In ruminants which have herbivorous diets, microorganisms in the gut produce another enzyme, cellulase to break down the cellulose cell walls of plant fiber.

Topic of Interest:
Ibn Sahl
Related Body of Text:
Ibn Sahl (c. 940-1000) was a Muslim mathematician, physicist and optics engineer of the Islamic Golden Age associated with the Abbasid court of Baghdad. Ibn Sahl's 984 treatise On Burning Mirrors and Lenses sets out his understanding of how curved mirrors and lenses bend and focus light. Ibn Sahl is credited with first discovering the law of refraction, usually called Snell's law.[1][2] He used the law of refraction to derive lens shapes that focus light with no geometric aberrations, known as anaclastic lenses. In the remaining parts of the treatise, Ibn Sahl dealt with parabolic mirrors, ellipsoidal mirrors, biconvex lenses, and techniques for drawing hyperbolic arcs. Ibn Sahl's treatise was used by Ibn al-Haitham (965-1039), one of the greatest Muslim scholars of optics.

Topic of Interest:
Wu Zetian
Related Body of Text:
Wu Zetian entered the Tang palace at 13 and became a concubine of Emperor Taizong. She did not become a favorite of Taizong's, and after his death in 649, she might have been expected to spend the rest of her life as a Buddhist nun, like his other childless concubines. However, through an unlikely fortuity - Empress Wang, the wife and empress of Emperor Taizong's son and successor Emperor Gaozong, wanted another beautiful concubine to divert Emperor Gaozong's favors from Consort Xiao, with whom Empress Wang was having a desperate struggle - Wang had her brought back to the palace and made a concubine of Emperor Gaozong. Consort Wu proceeded to defeat both Empress Wang and Consort Xiao in the struggle for Emperor Gaozong's affection, and subsequently, both Empress Wang and Consort Xiao were killed, and she was made empress.

Topic of Interest:
Apple Inc. logos
Related Body of Text:
Apple's first logo, designed by Jobs and Wayne, depicts Sir Isaac Newton sitting under an apple tree. Almost immediately, though, this was replaced by Rob Janoff's "rainbow Apple", the now-familiar rainbow-colored silhouette of an apple with a bite taken out of it. Janoff presented Jobs with several different monochromatic themes for the "bitten" logo,

and Jobs immediately took a liking to it. While Jobs liked the logo, he insisted it be in color to humanize the company.[128][129] The Apple logo was designed with a bite so that it would be recognized as an apple rather than a cherry. The colored stripes were conceived to make the logo more accessible, and to represent the fact the monitor could reproduce images in color.[130] In 1998, with the roll-out of the new iMac, Apple discontinued the rainbow theme and began to use monochromatic themes, nearly identical in shape to its previous rainbow incarnation.

Topic of Interest:
A prince at the seaside
Related Body of Text:
The Prince of Wales first visited Brighton (short for Brighthelmstone) in 1783, aged 21, staying with his uncle at Grove House on the Steine (or Steyne), a broad street that led from the seafront into the heart of town. He was prompted partly by his ever-lively desire to escape the disapproving eyes of his father's court, and partly by the recommendation of his physicians, who suggested that sea water might ease the glandular swellings in his neck. This sea-water cure had been the original cause of the rise in the popularity of Brighton as a watering place, which had started around 1765, courtesy of a Dr Richard Russell of Lewes who had publicized the health-giving properties of bathing in, and drinking, sea water in his A Dissertation: Concerning the Use of Sea Water in Diseases of the Glands, etc. (1752). Sea water taken one way or another, according to Russell, would cure almost any disease, including 'fluxions of redundant humours', rheumatism, madness, consumption, impotence, rabies and childish ailments.

Appendix D

Segments of Specific Question Roles Written in Perl Language

In order to transform tagged sentences into specific questions we apply 350 rules written in Perl language. Below, you can find the whole code. As you can see, for example, in the cases when the arguments are about time (ARGM-TMP) or about locations (ARGM-LOC) the question words of the specific questions should be *When* and *Where*, respectively.

```
#!/usr/bin/perl
$| = 1;
$/="\n";
$inputFile='Wu-Zetian.txt' or die "cannnot open";
open(IN, $inputFile);
$outFile= "../QG/SemQuestions/Wu-Zetian.txt";
open(OUT, ">$outFile");
while(<IN>)
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARG0.*?\])/)
        {       $_=~s/(\[ARG0.*?\])/Who/;
                #print $_;
                if($_ =~ m/(\n)/)
                {       $_=~s/(\n)/?\n/;}
                if($_=~ m/\d+:/)
                {$_=~s/\d+://;}
                if($_=~ m/\[R-ARG\d/)
                {
                        $_=~ s/\[R-ARG\d//g;
                }
                if($_=~ m/\[R-ARGM-LOC/)
                {
                        $_=~ s/\[R-ARGM-LOC//g;
                }
                if($_=~ m/\[C-ARG\d/)
                {
```

```
        $_=~ s/\[C-ARG\d//g;
}
if($_=~ m/\[TARGET/)
{
        $_=~ s/\[TARGET//g;
}
if($_=~ m/\[ARG0/)
{
        $_=~ s/\[ARG0//g;
}
if($_=~ m/\[ARG1/)
{
        $_=~ s/\[ARG1//g;
}
if($_=~ m/\[ARG2/)
{
        $_=~ s/\[ARG2//g;
}
if($_=~ m/\[ARGM-MNR/)
{
        $_=~ s/\[ARGM-MNR//g;
}
if($_=~ m/\[ARGM-CAU/)
{
        $_=~ s/\[ARGM-CAU//g;
}
if($_=~ m/\[ARGM-PNC/)
{
        $_=~ s/\[ARGM-PNC//g;
}
if($_=~ m/\[ARGM-TMP/)
{
        $_=~ s/\[ARGM-TMP//g;
}
if($_=~ m/\[ARGM-LOC/)
{
        $_=~ s/\[ARGM-LOC//g;
}
if($_=~ m/\[ARGM-DIS/)
{
        $_=~ s/\[ARGM-DIS//g;
}
if($_=~ m/\[ARGM-NEG/)
{
        $_=~ s/\[ARGM-NEG//g;
```

```
                }
                if($_=~ m/\[ARGM-DIR/)
                {
                        $_=~ s/\[ARGM-DIR//g;
                }
                if($_=~ m/\[ARGM-MOD/)
                {
                        $_=~ s/\[ARGM-MOD//g;
                }
                if($_=~ m/\[ARGM-ADV/)
                {
                        $_=~ s/\[ARGM-ADV//g;
                }
                if($_=~ m/\[ARG3/)
                {
                        $_=~ s/\[ARG3//g;
                }
                if($_=~ m/\[ARG4/)
                {
                        $_=~ s/\[ARG4//g;
                }
                if($_=~ m/\[ARGM-EXT/)
                {
                        $_=~ s/\[ARGM-EXT//g;
                }
                if($_=~ m/\[ARGM-PRD/)
                {
                        $_=~ s/\[ARGM-PRD//g;
                }
                if($_=~ m/\]/)
                {
                        $_=~ s/\]//g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
open(IN, $inputFile);
while(<IN>)
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARG1.*?\])/)
        {       $_=~s/(\[ARG1.*?\])/What/;
```

```
#print $_;
if($_ =~ m/(\n)/)
{        $_=~s/(\n)/?\n/;}
if($_=~ m/\d+:/)
{$_=~s/\d+://;}
if($_=~ m/\[R-ARG\d/)
{
         $_=~ s/\[R-ARG\d//g;
}
if($_=~ m/\[R-ARGM-LOC/)
{
         $_=~ s/\[R-ARGM-LOC//g;
}
                    if($_=~ m/\[C-ARG\d/)
{
         $_=~ s/\[C-ARG\d//g;
}
if($_=~ m/\[TARGET/)
{
         $_=~ s/\[TARGET//g;
}
if($_=~ m/\[ARG0/)
{
         $_=~ s/\[ARG0//g;
}
if($_=~ m/\[ARG1/)
{
         $_=~ s/\[ARG1//g;
}
if($_=~ m/\[ARG2/)
{
         $_=~ s/\[ARG2//g;
}
if($_=~ m/\[ARGM-MNR/)
{
         $_=~ s/\[ARGM-MNR//g;
}
if($_=~ m/\[ARGM-CAU/)
{
         $_=~ s/\[ARGM-CAU//g;
}
if($_=~ m/\[ARGM-PNC/)
{
         $_=~ s/\[ARGM-PNC//g;
}
```

```
if ($_=~ m/\[ARGM-TMP/)
{
        $_=~ s/\[ARGM-TMP//g;
}
if ($_=~ m/\[ARGM-LOC/)
{
        $_=~ s/\[ARGM-LOC//g;
}
if ($_=~ m/\[ARGM-DIS/)
{
        $_=~ s/\[ARGM-DIS//g;
}
if ($_=~ m/\[ARGM-NEG/)
{
        $_=~ s/\[ARGM-NEG//g;
}
if ($_=~ m/\[ARGM-DIR/)
{
        $_=~ s/\[ARGM-DIR//g;
}
if ($_=~ m/\[ARGM-MOD/)
{
        $_=~ s/\[ARGM-MOD//g;
}
if ($_=~ m/\[ARGM-ADV/)
{
        $_=~ s/\[ARGM-ADV//g;
}
                if ($_=~ m/\[ARG3/)
{
        $_=~ s/\[ARG3//g;
}
if ($_=~ m/\[ARG4/)
{
        $_=~ s/\[ARG4//g;
}
if ($_=~ m/\[ARGM-EXT/)
{
        $_=~ s/\[ARGM-EXT//g;
}
if ($_=~ m/\[ARGM-PRD/)
{
        $_=~ s/\[ARGM-PRD//g;
}
if ($_=~ m/\]/)
```

```
                {
                        $_=~ s/\]//g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
open(IN, $inputFile);
while(<IN>)
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARG2.*?\])/)
        {     $_=~s/(\[ARG2.*?\])/What/;
                #print $_;
                if($_ =~ m/(\n)/)
                {        $_=~s/(\n)/?\n/;}
                if($_=~ m/\d+:/)
                {$_=~s/\d+://;}
                if($_=~ m/\[R-ARG\d/)
                {
                        $_=~ s/\[R-ARG\d//g;
                }
                if($_=~ m/\[R-ARGM-LOC/)
                {
                        $_=~ s/\[R-ARGM-LOC//g;
                }
                if($_=~ m/\[C-ARG\d/)
                {
                        $_=~ s/\[C-ARG\d//g;
                }
                if($_=~ m/\[TARGET/)
                {
                        $_=~ s/\[TARGET//g;
                }
                if($_=~ m/\[ARG0/)
                {
                        $_=~ s/\[ARG0//g;
                }
                if($_=~ m/\[ARG1/)
                {
                        $_=~ s/\[ARG1//g;
                }
                if($_=~ m/\[ARG2/)
```

```
{
        $_=~ s/\[ARG2//g;
}
if ($_=~ m/\[ARGM-MNR/)
{
        $_=~ s/\[ARGM-MNR//g;
}
if ($_=~ m/\[ARGM-CAU/)
{
        $_=~ s/\[ARGM-CAU//g;
}
if ($_=~ m/\[ARGM-PNC/)
{
        $_=~ s/\[ARGM-PNC//g;
}
if ($_=~ m/\[ARGM-TMP/)
{
        $_=~ s/\[ARGM-TMP//g;
}
if ($_=~ m/\[ARGM-LOC/)
{
        $_=~ s/\[ARGM-LOC//g;
}
if ($_=~ m/\[ARGM-DIS/)
{
        $_=~ s/\[ARGM-DIS//g;
}
if ($_=~ m/\[ARGM-NEG/)
{
        $_=~ s/\[ARGM-NEG//g;
}
if ($_=~ m/\[ARGM-DIR/)
{
        $_=~ s/\[ARGM-DIR//g;
}
if ($_=~ m/\[ARGM-MOD/)
{
        $_=~ s/\[ARGM-MOD//g;
}
if ($_=~ m/\[ARGM-ADV/)
{
        $_=~ s/\[ARGM-ADV//g;
}
                if ($_=~ m/\[ARG3/)
{
```

```
                                $_=~ s/\[ARG3//g;
                        }
                        if($_=~ m/\[ARG4/)
                        {
                                $_=~ s/\[ARG4//g;
                        }
                        if($_=~ m/\[ARGM-EXT/)
                        {
                                $_=~ s/\[ARGM-EXT//g;
                        }
                        if($_=~ m/\[ARGM-PRD/)
                        {
                                $_=~ s/\[ARGM-PRD//g;
                        }
                        if($_=~ m/\]/)
                        {
                                $_=~ s/\]//g;
                        }
                        print $_;
                        print OUT $_;
                }
                close(IN, $inputFile);
                #last;
}
open(IN, $inputFile);
while(<IN>)
{
                #print $_."\n\n\n";
                if($_ =~ m/(\[ARGM-MNR.*?\])/)
                {       $_=~s/(\[ARGM-MNR.*?\])/How/;
                        #print $_;
                        if($_ =~ m/(\n)/)
                        {       $_=~s/(\n)/?\n/;}
                        if($_=~ m/\d+:/)
                        {$_=~s/\d+://;}
                        if($_=~ m/\[R-ARG\d/)
                        {
                                $_=~ s/\[R-ARG\d//g;
                        }
                        if($_=~ m/\[R-ARGM-LOC/)
                        {
                                $_=~ s/\[R-ARGM-LOC//g;
                        }
                        if($_=~ m/\[C-ARG\d/)
                        {
```

```
                $_=~ s/\[C-ARG\d//g;
        }
        if($_=~ m/\[TARGET/)
        {
                $_=~ s/\[TARGET//g;
        }
        if($_=~ m/\[ARG0/)
        {
                $_=~ s/\[ARG0//g;
        }
        if($_=~ m/\[ARG1/)
        {
                $_=~ s/\[ARG1//g;
        }
        if($_=~ m/\[ARG2/)
        {
                $_=~ s/\[ARG2//g;
        }
        if($_=~ m/\[ARGM-MNR/)
        {
                $_=~ s/\[ARGM-MNR//g;
        }
        if($_=~ m/\[ARGM-CAU/)
        {
                $_=~ s/\[ARGM-CAU//g;
        }
        if($_=~ m/\[ARGM-PNC/)
        {
                $_=~ s/\[ARGM-PNC//g;
        }
        if($_=~ m/\[ARGM-TMP/)
        {
                $_=~ s/\[ARGM-TMP//g;
        }
        if($_=~ m/\[ARGM-LOC/)
        {
                $_=~ s/\[ARGM-LOC//g;
        }
        if($_=~ m/\[ARGM-DIS/)
        {
                $_=~ s/\[ARGM-DIS//g;
        }
        if($_=~ m/\[ARGM-NEG/)
        {
                $_=~ s/\[ARGM-NEG//g;
                }
                if($_=~ m/\[ARGM-DIR/)
                {
                        $_=~ s/\[ARGM-DIR//g;
                }
                if($_=~ m/\[ARGM-MOD/)
                {
                        $_=~ s/\[ARGM-MOD//g;
                }
                if($_=~ m/\[ARGM-ADV/)
                {
                        $_=~ s/\[ARGM-ADV//g;
                }
                                if($_=~ m/\[ARG3/)
                {
                        $_=~ s/\[ARG3//g;
                }
                if($_=~ m/\[ARG4/)
                {
                        $_=~ s/\[ARG4//g;
                }
                if($_=~ m/\[ARGM-EXT/)
                {
                        $_=~ s/\[ARGM-EXT//g;
                }
                if($_=~ m/\[ARGM-PRD/)
                {
                        $_=~ s/\[ARGM-PRD//g;
                }
                if($_=~ m/\]/)
                {
                        $_=~ s/\]//g;
                }
                if($_=~ m/(.*)How/)
                {
                        $_=~ s/(.*)How/How$1/g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
open(IN, $inputFile);
while(<IN>)
```

```
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARGM-CAU.*?\])/)
        {     $_=~s/(\[ARGM-CAU.*?\])/Why/;
                #print $_;
                if($_ =~ m/(\n)/)
                {         $_=~s/(\n)/?\n/;}
                if($_=~ m/\d+:/)
                {$_=~s/\d+://;}
                if($_=~ m/\[R-ARG\d/)
                {
                        $_=~ s/\[R-ARG\d//g;
                }
                if($_=~ m/\[R-ARGM-LOC/)
                {
                        $_=~ s/\[R-ARGM-LOC//g;
                }
                if($_=~ m/\[C-ARG\d/)
                {
                        $_=~ s/\[C-ARG\d//g;
                }
                if($_=~ m/\[TARGET/)
                {
                        $_=~ s/\[TARGET//g;
                }
                if($_=~ m/\[ARG0/)
                {
                        $_=~ s/\[ARG0//g;
                }
                if($_=~ m/\[ARG1/)
                {
                        $_=~ s/\[ARG1//g;
                }
                if($_=~ m/\[ARG2/)
                {
                        $_=~ s/\[ARG2//g;
                }
                if($_=~ m/\[ARGM-MNR/)
                {
                        $_=~ s/\[ARGM-MNR//g;
                }
                if($_=~ m/\[ARGM-CAU/)
                {
                        $_=~ s/\[ARGM-CAU//g;
                }
```

```
if($_=~ m/\[ARGM-PNC/)
{
        $_=~ s/\[ARGM-PNC//g;
}
if($_=~ m/\[ARGM-TMP/)
{
        $_=~ s/\[ARGM-TMP//g;
}
if($_=~ m/\[ARGM-LOC/)
{
        $_=~ s/\[ARGM-LOC//g;
}
if($_=~ m/\[ARGM-DIS/)
{
        $_=~ s/\[ARGM-DIS//g;
}
if($_=~ m/\[ARGM-NEG/)
{
        $_=~ s/\[ARGM-NEG//g;
}
if($_=~ m/\[ARGM-DIR/)
{
        $_=~ s/\[ARGM-DIR//g;
}
if($_=~ m/\[ARGM-MOD/)
{
        $_=~ s/\[ARGM-MOD//g;
}
if($_=~ m/\[ARGM-ADV/)
{
        $_=~ s/\[ARGM-ADV//g;
}
                if($_=~ m/\[ARG3/)
{
        $_=~ s/\[ARG3//g;
}
if($_=~ m/\[ARG4/)
{
        $_=~ s/\[ARG4//g;
}
if($_=~ m/\[ARGM-EXT/)
{
        $_=~ s/\[ARGM-EXT//g;
}
if($_=~ m/\[ARGM-PRD/)
```

```
                {
                        $_=~ s/\[ARGM-PRD//g;
                }
                if($_=~ m/\]/)
                {
                        $_=~ s/\]//g;
                }
                if($_=~ m/(.*)Why/)
                {
                        $_=~ s/(.*)Why/Why$1/g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
open(IN, $inputFile);
while(<IN>)
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARGM-PNC.*?\])/)
        {     $_=~s/(\[ARGM-PNC.*?\])/Why/;
                #print $_;
                if($_ =~ m/(\n)/)
                {       $_=~s/(\n)/?\n/;}
                if($_=~ m/\d+:/)
                {$_=~s/\d+://;}
                if($_=~ m/\[R-ARG\d/)
                {
                        $_=~ s/\[R-ARG\d//g;
                }
                if($_=~ m/\[R-ARGM-LOC/)
                {
                        $_=~ s/\[R-ARGM-LOC//g;
                }
        if($_=~ m/\[C-ARG\d/)
                {
                        $_=~ s/\[C-ARG\d//g;
                }
                if($_=~ m/\[TARGET/)
                {
                        $_=~ s/\[TARGET//g;
                }
                if($_=~ m/\[ARG0/)
```

```
{
        $_=~ s/\[ARG0//g;
}
if ($_=~ m/\[ARG1/)
{
        $_=~ s/\[ARG1//g;
}
if ($_=~ m/\[ARG2/)
{
        $_=~ s/\[ARG2//g;
}
if ($_=~ m/\[ARGM-MNR/)
{
        $_=~ s/\[ARGM-MNR//g;
}
if ($_=~ m/\[ARGM-CAU/)
{
        $_=~ s/\[ARGM-CAU//g;
}
if ($_=~ m/\[ARGM-PNC/)
{
        $_=~ s/\[ARGM-PNC//g;
}
if ($_=~ m/\[ARGM-TMP/)
{
        $_=~ s/\[ARGM-TMP//g;
}
if ($_=~ m/\[ARGM-LOC/)
{
        $_=~ s/\[ARGM-LOC//g;
}
if ($_=~ m/\[ARGM-DIS/)
{
        $_=~ s/\[ARGM-DIS//g;
}
if ($_=~ m/\[ARGM-NEG/)
{
        $_=~ s/\[ARGM-NEG//g;
}
if ($_=~ m/\[ARGM-DIR/)
{
        $_=~ s/\[ARGM-DIR//g;
}
if ($_=~ m/\[ARGM-MOD/)
{
```

```
                    $_=~ s/\[ARGM-MOD//g;
                }
                if($_=~ m/\[ARGM-ADV/)
                {
                    $_=~ s/\[ARGM-ADV//g;
                }
                                if($_=~ m/\[ARG3/)
                {
                    $_=~ s/\[ARG3//g;
                }
                if($_=~ m/\[ARG4/)
                {
                    $_=~ s/\[ARG4//g;
                }
                if($_=~ m/\[ARGM-EXT/)
                {
                    $_=~ s/\[ARGM-EXT//g;
                }
                if($_=~ m/\[ARGM-PRD/)
                {
                    $_=~ s/\[ARGM-PRD//g;
                }
                if($_=~ m/\]/)
                {
                    $_=~ s/\]//g;
                }
                if($_=~ m/(.*)Why/)
                {
                    $_=~ s/(.*)Why/Why$1/g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
open(IN, $inputFile);
while(<IN>)
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARGM-TMP.*?\])/)
        {    $_=~s/(\[ARGM-TMP.*?\])/When/;
                #print $_;
                if($_ =~ m/(\n)/)
                {       $_=~s/(\n)/?\n/;}
```

```
if ($_=~ m/\d+:/)
{$_=~s/\d+://;}
if ($_=~ m/\[R-ARG\d/)
{
          $_=~ s/\[R-ARG\d//g;
}
if ($_=~ m/\[R-ARGM-LOC/)
{
          $_=~ s/\[R-ARGM-LOC//g;
}
if ($_=~ m/\[C-ARG\d/)
{
          $_=~ s/\[C-ARG\d//g;
}
if ($_=~ m/\[TARGET/)
{
          $_=~ s/\[TARGET//g;
}
if ($_=~ m/\[ARG0/)
{
          $_=~ s/\[ARG0//g;
}
if ($_=~ m/\[ARG1/)
{
          $_=~ s/\[ARG1//g;
}
if ($_=~ m/\[ARG2/)
{
          $_=~ s/\[ARG2//g;
}
if ($_=~ m/\[ARGM-MNR/)
{
          $_=~ s/\[ARGM-MNR//g;
}
if ($_=~ m/\[ARGM-CAU/)
{
          $_=~ s/\[ARGM-CAU//g;
}
if ($_=~ m/\[ARGM-PNC/)
{
          $_=~ s/\[ARGM-PNC//g;
}
if ($_=~ m/\[ARGM-TMP/)
{
          $_=~ s/\[ARGM-TMP//g;
```

```
}
if ($_=~ m/\[ARGM-LOC/)
{
          $_=~ s/\[ARGM-LOC//g;
}
if ($_=~ m/\[ARGM-DIS/)
{
          $_=~ s/\[ARGM-DIS//g;
}
if ($_=~ m/\[ARGM-NEG/)
{
          $_=~ s/\[ARGM-NEG//g;
}
if ($_=~ m/\[ARGM-DIR/)
{
          $_=~ s/\[ARGM-DIR//g;
}
if ($_=~ m/\[ARGM-MOD/)
{
          $_=~ s/\[ARGM-MOD//g;
}
if ($_=~ m/\[ARGM-ADV/)
{
          $_=~ s/\[ARGM-ADV//g;
}
                    if ($_=~ m/\[ARG3/)
{
          $_=~ s/\[ARG3//g;
}
if ($_=~ m/\[ARG4/)
{
          $_=~ s/\[ARG4//g;
}
if ($_=~ m/\[ARGM-EXT/)
{
          $_=~ s/\[ARGM-EXT//g;
}
if ($_=~ m/\[ARGM-PRD/)
{
          $_=~ s/\[ARGM-PRD//g;
}
if ($_=~ m/\]/)
{
          $_=~ s/\]//g;
}
```

```
                if($_=~ m/(.*)When/)
                {
                        $_=~ s/(.*)When/When$1/g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
open(IN, $inputFile);
while(<IN>)
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARGM-LOC.*?\])/)
        {     $_=~s/(\[ARGM-LOC.*?\])/Where/;
                #print $_;
                if($_ =~ m/(\n)/)
                {       $_=~s/(\n)/?\n/;}
                if($_=~ m/\d+:/)
                {$_=~s/\d+://;}
                if($_=~ m/\[R-ARG\d/)
                {
                        $_=~ s/\[R-ARG\d//g;
                }
                if($_=~ m/\[R-ARGM-LOC/)
                {
                        $_=~ s/\[R-ARGM-LOC//g;
                }
                if($_=~ m/\[C-ARG\d/)
                {
                        $_=~ s/\[C-ARG\d//g;
                }
                if($_=~ m/\[TARGET/)
                {
                        $_=~ s/\[TARGET//g;
                }
                if($_=~ m/\[ARG0/)
                {
                        $_=~ s/\[ARG0//g;
                }
                if($_=~ m/\[ARG1/)
                {
                        $_=~ s/\[ARG1//g;
                }
```

```
if ($_=~ m/\[ARG2/)
{
        $_=~ s/\[ARG2//g;
}
if ($_=~ m/\[ARGM-MNR/)
{
        $_=~ s/\[ARGM-MNR//g;
}
if ($_=~ m/\[ARGM-CAU/)
{
        $_=~ s/\[ARGM-CAU//g;
}
if ($_=~ m/\[ARGM-PNC/)
{
        $_=~ s/\[ARGM-PNC//g;
}
if ($_=~ m/\[ARGM-TMP/)
{
        $_=~ s/\[ARGM-TMP//g;
}
if ($_=~ m/\[ARGM-LOC/)
{
        $_=~ s/\[ARGM-LOC//g;
}
if ($_=~ m/\[ARGM-DIS/)
{
        $_=~ s/\[ARGM-DIS//g;
}
if ($_=~ m/\[ARGM-NEG/)
{
        $_=~ s/\[ARGM-NEG//g;
}
if ($_=~ m/\[ARGM-DIR/)
{
        $_=~ s/\[ARGM-DIR//g;
}
if ($_=~ m/\[ARGM-MOD/)
{
        $_=~ s/\[ARGM-MOD//g;
}
if ($_=~ m/\[ARGM-ADV/)
{
        $_=~ s/\[ARGM-ADV//g;
}
                if ($_=~ m/\[ARG3/)
```

```
                    {
                            $_=~ s/\[ARG3//g;
                    }
                    if($_=~ m/\[ARG4/)
                    {
                            $_=~ s/\[ARG4//g;
                    }
                    if($_=~ m/\[ARGM-EXT/)
                    {
                            $_=~ s/\[ARGM-EXT//g;
                    }
                    if($_=~ m/\[ARGM-PRD/)
                    {
                            $_=~ s/\[ARGM-PRD//g;
                    }
                    if($_=~ m/\]/)
                    {
                            $_=~ s/\]//g;
                    }
                    if($_=~ m/(.*)Where/)
                    {
                            $_=~ s/(.*)Where/Where$1/g;
                    }
                    print $_;
                    print OUT $_;
          }
          close(IN, $inputFile);
          #last;
}
open(IN, $inputFile);
while(<IN>)
{
          #print $_."\n\n\n";
          if($_ =~ m/(\[ARGM-DIS.*?\])/)
          {      $_=~s/(\[ARGM-DIS.*?\])/How/;
                    #print $_;
                    if($_ =~ m/(\n)/)
                    {         $_=~s/(\n)/?\n/;}
                    if($_=~ m/\d+:/)
                    {$_=~s/\d+://;}
                    if($_=~ m/\[R-ARG\d/)
                    {
                            $_=~ s/\[R-ARG\d//g;
                    }
                    if($_=~ m/\[R-ARGM-LOC/)
```

```
{
        $_=~ s/\[R-ARGM-LOC//g;
}
if ($_=~ m/\[C-ARG\d/)
{
        $_=~ s/\[C-ARG\d//g;
}
if ($_=~ m/\[TARGET/)
{
        $_=~ s/\[TARGET//g;
}
if ($_=~ m/\[ARG0/)
{
        $_=~ s/\[ARG0//g;
}
if ($_=~ m/\[ARG1/)
{
        $_=~ s/\[ARG1//g;
}
if ($_=~ m/\[ARG2/)
{
        $_=~ s/\[ARG2//g;
}
if ($_=~ m/\[ARGM-MNR/)
{
        $_=~ s/\[ARGM-MNR//g;
}
if ($_=~ m/\[ARGM-CAU/)
{
        $_=~ s/\[ARGM-CAU//g;
}
if ($_=~ m/\[ARGM-PNC/)
{
        $_=~ s/\[ARGM-PNC//g;
}
if ($_=~ m/\[ARGM-TMP/)
{
        $_=~ s/\[ARGM-TMP//g;
}
if ($_=~ m/\[ARGM-LOC/)
{
        $_=~ s/\[ARGM-LOC//g;
}
if ($_=~ m/\[ARGM-DIS/)
{
```

```
                $_=~ s/\[ARGM-DIS//g;
        }
        if($_=~ m/\[ARGM-NEG/)
        {
                $_=~ s/\[ARGM-NEG//g;
        }
        if($_=~ m/\[ARGM-DIR/)
        {
                $_=~ s/\[ARGM-DIR//g;
        }
        if($_=~ m/\[ARGM-MOD/)
        {
                $_=~ s/\[ARGM-MOD//g;
        }
        if($_=~ m/\[ARGM-ADV/)
        {
                $_=~ s/\[ARGM-ADV//g;
        }
                        if($_=~ m/\[ARG3/)
        {
                $_=~ s/\[ARG3//g;
        }
        if($_=~ m/\[ARG4/)
        {
                $_=~ s/\[ARG4//g;
        }
        if($_=~ m/\[ARGM-EXT/)
        {
                $_=~ s/\[ARGM-EXT//g;
        }
        if($_=~ m/\[ARGM-PRD/)
        {
                $_=~ s/\[ARGM-PRD//g;
        }
        if($_=~ m/\]/)
        {
                $_=~ s/\]//g;
        }
        if($_=~ m/(.*)How/)
        {
                $_=~ s/(.*)How/How$1/g;
        }
        print $_;
        print OUT $_;
}
```

```
          close(IN, $inputFile);
          #last;
}
open(IN, $inputFile);
while(<IN>)
{
          #print $_."\n\n\n";
          if($_ =~ m/(\[ARGM-ADV.*?\])/)
          {      $_=~s/(\[ARGM-ADV.*?\])/In what circumstances/;
                    #print $_;
                    if($_ =~ m/(\n)/)
                    {         $_=~s/(\n)/?\n/;}
                    if($_=~ m/\d+:/)
                    {$_=~s/\d+://;}
                    if($_=~ m/\[R-ARG\d/)
                    {
                              $_=~ s/\[R-ARG\d//g;
                    }
                    if($_=~ m/\[R-ARGM-LOC/)
                    {
                              $_=~ s/\[R-ARGM-LOC//g;
                    }
                    if($_=~ m/\[C-ARG\d/)
                    {
                              $_=~ s/\[C-ARG\d//g;
                    }
                    if($_=~ m/\[TARGET/)
                    {
                              $_=~ s/\[TARGET//g;
                    }
                    if($_=~ m/\[ARG0/)
                    {
                              $_=~ s/\[ARG0//g;
                    }
                    if($_=~ m/\[ARG1/)
                    {
                              $_=~ s/\[ARG1//g;
                    }
                    if($_=~ m/\[ARG2/)
                    {
                              $_=~ s/\[ARG2//g;
                    }
                    if($_=~ m/\[ARGM-MNR/)
                    {
                              $_=~ s/\[ARGM-MNR//g;
```

```
}
if ($_=~ m/\[ARGM-CAU/)
{
        $_=~ s/\[ARGM-CAU//g;
}
if ($_=~ m/\[ARGM-PNC/)
{
        $_=~ s/\[ARGM-PNC//g;
}
if ($_=~ m/\[ARGM-TMP/)
{
        $_=~ s/\[ARGM-TMP//g;
}
if ($_=~ m/\[ARGM-LOC/)
{
        $_=~ s/\[ARGM-LOC//g;
}
if ($_=~ m/\[ARGM-DIS/)
{
        $_=~ s/\[ARGM-DIS//g;
}
if ($_=~ m/\[ARGM-NEG/)
{
        $_=~ s/\[ARGM-NEG//g;
}
if ($_=~ m/\[ARGM-DIR/)
{
        $_=~ s/\[ARGM-DIR//g;
}
if ($_=~ m/\[ARGM-MOD/)
{
        $_=~ s/\[ARGM-MOD//g;
}
if ($_=~ m/\[ARGM-ADV/)
{
        $_=~ s/\[ARGM-ADV//g;
}
                if ($_=~ m/\[ARG3/)
{
        $_=~ s/\[ARG3//g;
}
if ($_=~ m/\[ARG4/)
{
        $_=~ s/\[ARG4//g;
}
```

```
                if($_=~ m/\[ARGM-EXT/)
                {
                        $_=~ s/\[ARGM-EXT//g;
                }
                if($_=~ m/\[ARGM-PRD/)
                {
                        $_=~ s/\[ARGM-PRD//g;
                }
                if($_=~ m/\]/)
                {
                        $_=~ s/\]//g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
open(IN, $inputFile);
while(<IN>)
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARG3.*?\])/)
        {     $_=~s/(\[ARG3.*?\])/Where/;
                #print $_;
                if($_ =~ m/(\n)/)
                {       $_=~s/(\n)/?\n/;}
                if($_=~ m/\d+:/)
                {$_=~s/\d+://;}
                if($_=~ m/\[R-ARG\d/)
                {
                        $_=~ s/\[R-ARG\d//g;
                }
                if($_=~ m/\[R-ARGM-LOC/)
                {
                        $_=~ s/\[R-ARGM-LOC//g;
                }
        if($_=~ m/\[C-ARG\d/)
                {
                        $_=~ s/\[C-ARG\d//g;
                }
                if($_=~ m/\[TARGET/)
                {
                        $_=~ s/\[TARGET//g;
                }
```

```
if ($_=~ m/\[ARG0/)
{
          $_=~ s/\[ARG0//g;
}
if ($_=~ m/\[ARG1/)
{
          $_=~ s/\[ARG1//g;
}
if ($_=~ m/\[ARG2/)
{
          $_=~ s/\[ARG2//g;
}
if ($_=~ m/\[ARGM-MNR/)
{
          $_=~ s/\[ARGM-MNR//g;
}
if ($_=~ m/\[ARGM-CAU/)
{
          $_=~ s/\[ARGM-CAU//g;
}
if ($_=~ m/\[ARGM-PNC/)
{
          $_=~ s/\[ARGM-PNC//g;
}
if ($_=~ m/\[ARGM-TMP/)
{
          $_=~ s/\[ARGM-TMP//g;
}
if ($_=~ m/\[ARGM-LOC/)
{
          $_=~ s/\[ARGM-LOC//g;
}
if ($_=~ m/\[ARGM-DIS/)
{
          $_=~ s/\[ARGM-DIS//g;
}
if ($_=~ m/\[ARGM-NEG/)
{
          $_=~ s/\[ARGM-NEG//g;
}
if ($_=~ m/\[ARGM-DIR/)
{
          $_=~ s/\[ARGM-DIR//g;
}
if ($_=~ m/\[ARGM-MOD/)
```

```
                {
                        $_=~ s/\[ARGM-MOD//g;
                }
                if($_=~ m/\[ARGM-ADV/)
                {
                        $_=~ s/\[ARGM-ADV//g;
                }
                                if($_=~ m/\[ARG3/)
                {
                        $_=~ s/\[ARG3//g;
                }
                if($_=~ m/\[ARG4/)
                {
                        $_=~ s/\[ARG4//g;
                }
                if($_=~ m/\[ARGM-EXT/)
                {
                        $_=~ s/\[ARGM-EXT//g;
                }
                if($_=~ m/\[ARGM-PRD/)
                {
                        $_=~ s/\[ARGM-PRD//g;
                }
                if($_=~ m/\]/)
                {
                        $_=~ s/\]//g;
                }
                if($_=~ m/(.*)Where/)
                {
                        $_=~ s/(.*)Where/Where$1/g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
open(IN, $inputFile);
while(<IN>)
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARG4.*?\])/)
        {       $_=~s/(\[ARG4.*?\])/Where/;
                #print $_;
                if($_ =~ m/(\n)/)
```

```
{           $_=~s/(\n)/?\n/;}
if($_=~ m/\d+:/)
{$_=~s/\d+://;}
if($_=~ m/\[R–ARG\d/)
{
            $_=~ s/\[R–ARG\d//g;
}
if($_=~ m/\[R–ARGM–LOC/)
{
            $_=~ s/\[R–ARGM–LOC//g;
}
if($_=~ m/\[C–ARG\d/)
{
            $_=~ s/\[C–ARG\d//g;
}
if($_=~ m/\[TARGET/)
{
            $_=~ s/\[TARGET//g;
}
if($_=~ m/\[ARG0/)
{
            $_=~ s/\[ARG0//g;
}
if($_=~ m/\[ARG1/)
{
            $_=~ s/\[ARG1//g;
}
if($_=~ m/\[ARG2/)
{
            $_=~ s/\[ARG2//g;
}
if($_=~ m/\[ARGM–MNR/)
{
            $_=~ s/\[ARGM–MNR//g;
}
if($_=~ m/\[ARGM–CAU/)
{
            $_=~ s/\[ARGM–CAU//g;
}
if($_=~ m/\[ARGM–PNC/)
{
            $_=~ s/\[ARGM–PNC//g;
}
if($_=~ m/\[ARGM–TMP/)
{
```

```
            $_=~ s/\[ARGM-TMP//g;
}
if($_=~ m/\[ARGM-LOC/)
{
            $_=~ s/\[ARGM-LOC//g;
}
if($_=~ m/\[ARGM-DIS/)
{
            $_=~ s/\[ARGM-DIS//g;
}
if($_=~ m/\[ARGM-NEG/)
{
            $_=~ s/\[ARGM-NEG//g;
}
if($_=~ m/\[ARGM-DIR/)
{
            $_=~ s/\[ARGM-DIR//g;
}
if($_=~ m/\[ARGM-MOD/)
{
            $_=~ s/\[ARGM-MOD//g;
}
if($_=~ m/\[ARGM-ADV/)
{
            $_=~ s/\[ARGM-ADV//g;
}
                        if($_=~ m/\[ARG3/)
{
            $_=~ s/\[ARG3//g;
}
if($_=~ m/\[ARG4/)
{
            $_=~ s/\[ARG4//g;
}
if($_=~ m/\[ARG4/)
{
            $_=~ s/\[ARG4//g;
}
if($_=~ m/\[ARGM-EXT/)
{
            $_=~ s/\[ARGM-EXT//g;
}
if($_=~ m/\[ARGM-PRD/)
{
            $_=~ s/\[ARGM-PRD//g;
```

```
                }
                if($_=~ m/\]/)
                {
                        $_=~ s/\]//g;
                }
                if($_=~ m/(.*)Where/)
                {
                        $_=~ s/(.*)Where/Where$1/g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
open(IN, $inputFile);
while(<IN>)
{
        #print $_."\n\n\n";
        if($_ =~ m/(\[ARGM-EXT.*?\])/)
        {     $_=~s/(\[ARGM-EXT.*?\])/To what extent/;
                #print $_;
                if($_ =~ m/(\n)/)
                {          $_=~s/(\n)/?\n/;}
                if($_=~ m/\d+:/)
                {$_=~s/\d+://;}
                if($_=~ m/\[R-ARG\d/)
                {
                        $_=~ s/\[R-ARG\d//g;
                }
                if($_=~ m/\[R-ARGM-LOC/)
                {
                        $_=~ s/\[R-ARGM-LOC//g;
                }
                if($_=~ m/\[C-ARG\d/)
                {
                        $_=~ s/\[C-ARG\d//g;
                }
                if($_=~ m/\[TARGET/)
                {
                        $_=~ s/\[TARGET//g;
                }
                if($_=~ m/\[ARG0/)
                {
                        $_=~ s/\[ARG0//g;
```

```
}
if ($_=~ m/\[ARG1/)
{
          $_=~ s/\[ARG1//g;
}
if ($_=~ m/\[ARG2/)
{
          $_=~ s/\[ARG2//g;
}
if ($_=~ m/\[ARGM-MNR/)
{
          $_=~ s/\[ARGM-MNR//g;
}
if ($_=~ m/\[ARGM-CAU/)
{
          $_=~ s/\[ARGM-CAU//g;
}
if ($_=~ m/\[ARGM-PNC/)
{
          $_=~ s/\[ARGM-PNC//g;
}
if ($_=~ m/\[ARGM-TMP/)
{
          $_=~ s/\[ARGM-TMP//g;
}
if ($_=~ m/\[ARGM-LOC/)
{
          $_=~ s/\[ARGM-LOC//g;
}
if ($_=~ m/\[ARGM-DIS/)
{
          $_=~ s/\[ARGM-DIS//g;
}
if ($_=~ m/\[ARGM-NEG/)
{
          $_=~ s/\[ARGM-NEG//g;
}
if ($_=~ m/\[ARGM-DIR/)
{
          $_=~ s/\[ARGM-DIR//g;
}
if ($_=~ m/\[ARGM-MOD/)
{
          $_=~ s/\[ARGM-MOD//g;
}
```

```
                if($_=~ m/\[ARGM-ADV/)
                {
                        $_=~ s/\[ARGM-ADV//g;
                }
                                if($_=~ m/\[ARG3/)
                {
                        $_=~ s/\[ARG3//g;
                }
                if($_=~ m/\[ARG4/)
                {
                        $_=~ s/\[ARG4//g;
                }
                if($_=~ m/\[ARGM-EXT/)
                {
                        $_=~ s/\[ARGM-EXT//g;
                }
                if($_=~ m/\[ARGM-PRD/)
                {
                        $_=~ s/\[ARGM-PRD//g;
                }
                if($_=~ m/\]/)
                {
                        $_=~ s/\]//g;
                }
                print $_;
                print OUT $_;
        }
        close(IN, $inputFile);
        #last;
}
```