Automatic Factual Question Generation from Text

Michael Heilman

CMU-LTI-11-004

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Vincent Aleven, Carnegie Mellon University
William W. Cohen, Carnegie Mellon University
Lori Levin, Carnegie Mellon University
Diane J. Litman, University of Pittsburgh
Noah A. Smith (chair), Carnegie Mellon University

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies

Abstract

Texts with potential educational value are becoming available through the Internet (e.g., Wikipedia, news services). However, using these new texts in classrooms introduces many challenges, one of which is that they usually lack practice exercises and assessments. Here, we address part of this challenge by automating the creation of a specific type of assessment item.

Specifically, we focus on automatically generating factual WH questions. Our goal is to create an automated system that can take as input a text and produce as output questions for assessing a reader's knowledge of the information in the text. The questions could then be presented to a teacher, who could select and revise the ones that he or she judges to be useful.

After introducing the problem, we describe some of the computational and linguistic challenges presented by factual question generation. We then present an implemented system that leverages existing natural language processing techniques to address some of these challenges. The system uses a combination of manually encoded transformation rules and a statistical question ranker trained on a tailored dataset of labeled system output.

We present experiments that evaluate individual components of the system as well as the system as a whole. We found, among other things, that the question ranker roughly doubled the acceptability rate of top-ranked questions.

In a user study, we tested whether K-12 teachers could efficiently create factual questions by selecting and revising suggestions from the system. Offering automatic suggestions reduced the time and effort spent by participants, though it also affected the types of questions that were created.

This research supports the idea that natural language processing can help teachers efficiently create instructional content. It provides solutions to some of the major challenges in question generation and an analysis and better understanding of those that remain.

Acknowledgements

In addition to the dissertation committee, the author would like to acknowledge the following people for helping him conduct this research: the reviewers of the publications related to this work, for their helpful comments; Brendan O'Connor, for developing the treeviz parse visualization tool, for his work on the ARKref tool, and for ideas about information extraction; Nathan Schneider, Dipanjan Das, Kevin Gimpel, Shay Cohen, and other members of the ARK research group, for various helpful discussions; Justin Betteridge for discussions about connections between question generation and information extraction; Nora Presson, Ruth Wylie, and Leigh Ann Sudol, for discussions about the user interface and user study; Maxine Eskenazi and Matthias Scheutz, for being his previous advisors; Howard Seltman, Nora Presson, and Tracy Sweet, for statistical advice; the organizers of the question generation workshops, for enabling many thoughtful discussions about the topic; Jack Mostow, for asking many good questions; and Jill Burstein, for providing links to some of the texts used in this work.

The author would also like to acknowledge partial support from the following organizations: the Siebel Scholars organization; the National Science Foundation, for a Graduate Research Fellowship awarded to the author and for grant IIS-0915187, awarded to Noah Smith; and the U.S. Department of Education's Institute of Education Sciences, for grant R305B040063 to Carnegie Mellon University and its Program for Interdisciplinary Education Research (PIER) directed by David Klahr and Sharon Carver. The views expressed in this work are those of the author and are not necessarily those of the above organizations.

Finally, the author would like to thank his parents and his wife for their unflagging support over many years.

Contents

Chapter 1

Introduction

How would someone tell whether you have read this text? They might ask you to summarize it or describe how it relates to your own research. They might ask you to discuss its strengths and weaknesses or to compare it to another paper.

Or, as a first step, just to see if you bothered to get through it, they might ask you what features we used in our ranking model, or what corpora we tested on. That is, at first, they might just ask you questions to check that you remember the basic facts. Then, if it is clear that you read more than the title and abstract, they might move on to more challenging questions.

Of course, you are probably a highly skilled and motivated reader, and there would be no need to assess whether you read and retained basic factual information. However, that is not the case with all readers. For example, an elementary school teacher might ask his or her students basic questions since they are still learning to read.

Generating such questions, and authoring reading assessments more generally, can be a time-consuming and effortful process. In this research, we work toward automating that process. In particular, we focus on the problem of automatically generating factual questions from individual texts.

We aim to create a system for question generation (QG) that can take as input an article of text (e.g., a web page or encyclopedia article that a teacher might select to supplement the materials in a textbook), and create as output a ranked list of factual questions. A user could then select and revise these questions in order to create practice exercises or part of a quiz to assess whether students read the text and retained knowledge about its topic.

We focus on QG about informational texts—that is, non-fiction texts that convey factual information rather than opinions. While QG about narratives and subjective essays would also be interesting and educationally relevant, we leave these problems to future work. To make our factual QG system generally useful, we avoid the use of domain-specific knowledge (e.g., about historical events or geographical locations) and instead focus on modeling fairly general lexical and syntactic phenomena related to questions and the presentation of factual information.

1.1 Illustrative Example of Factual Question Generation

In this section, we provide examples that illustrate that QG about explicit factual information is a challenging but still feasible task given current natural language processing (NLP) technologies.

We begin with a relatively straightforward example, taken from an *Encyclopedia Britannica Elementary Edition* article about the city of Monrovia.[1]

(1.1) . . . Monrovia was named after James Monroe, who was president of the United States in 1822. In that year a group of freed U.S. slaves, sponsored by a U.S. society, started a new settlement on the continent of their ancestors. As more settlers arrived from the United States and from the Caribbean area, the area they controlled grew larger. In 1847 Monrovia became the capital of the new country of Liberia. . . .

A number of acceptable factual questions can be generated from this sentence by analyzing its grammatical structure, labeling its lexical items with high-level semantic types (e.g., person, location, time), and then performing syntactic transformations such as subject-auxiliary inversion and WH-movement. From the first sentence, we can extract well-formed and specific questions such as the following:

(1.2) Who was president of the United States in 1822?

(1.3) When was James Monroe president of the United States?

(1.4) Who was Monrovia named after?

[1] We make use of a dataset of Encyclopedia Britannica texts from previous NLP research (Barzilay and Elhadad, 2003). We describe the dataset in more detail in Chapter 4.

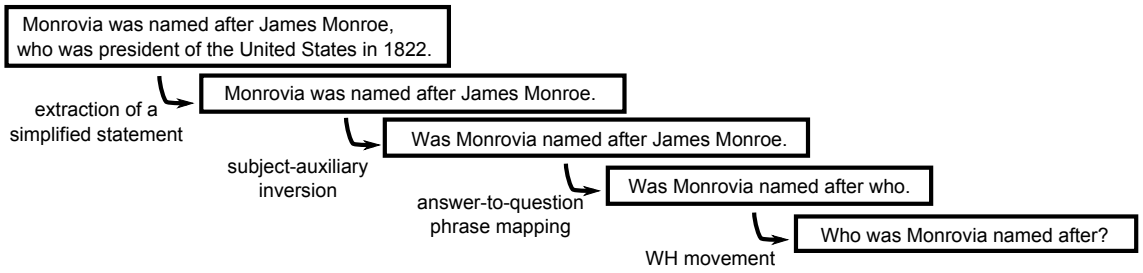

Figure 1.1: An illustration of how simple lexical and syntactic transformations can convert a statement into a question. Chapter 3 describes this process in detail.

(1.5) What was named after James Monroe?

Figure 1.1 is a rough illustration of the sequence of transformations used to generate question 1.4 (we describe each step in detail later in Chapter 3). We can also generate acceptable questions such as the following from the last sentence:

(1.6) When did Monrovia become the capital of the new country of Liberia?

(1.7) What did Monrovia become in 1847?

(1.8) What did Monrovia become the capital of in 1847?

(1.9) What became the capital of the new country of Liberia in 1847?

The second sentence is more challenging, leading to unacceptable outputs such as the following:

(1.10) What did a group of freed U. S. slaves start in that year?

(1.11) What started a new settlement on the continent of their ancestors in that year?

Question 1.10 demonstrates the challenge of asking questions by taking information out of a larger discourse. Specifically, the phrase *that year* cannot be resolved to 1822 in the absence of the preceding sentence. Question 1.11 shows the challenge of creating appropriate question phrases—in this case, identifying that the noun phrase headed by the word *group* should lead to a *who* question, or perhaps a question starting with the phrase *what group*.

We return to these two challenges, along with others, in Chapter 2. Also, in §3.4 of Chapter 3, we present a statistical ranking model, trained from labeled data, that will help us avoid bad questions such as 1.10 while preserving good questions such as 1.4.

1.2 Instructional Content Generation

Many classrooms rely almost exclusively on textbooks as a source of reading materials, either for teaching reading itself or for teaching other topics in areas such as social studies and science. While textbooks have many advantages (e.g., they are well-edited, they contain lesson plans and exercises), they are a relatively small and expensive resource. The texts and exercises in a particular textbook may not match the needs of learners in a particular classroom—either the cognitive needs of learners at various skill levels, or the motivational needs of learners with various interests. For example, a textbook may not provide sufficient materials for themed units or specialized curricula. A teacher in Pittsburgh might want to find outside texts for a unit on the history of the city of Pittsburgh. Or, a teacher at a charter school with an environmentally-themed curriculum might want to find texts about renewable energy and global warming.[2] Also, certain instructional methods, such as the "Reader's Workshop,"[3] make heavy use of outside texts.

In contrast to textbooks, the Internet provides a vast source of texts (e.g., from digital libraries, news sites, Wikipedia, etc.). In theory, this source of texts could provide materials that better match students' skills and interests—and thereby supplement the resources provided by textbooks.

There are two major challenges, however, in making use of new texts:

- First, the Internet is not designed as an educational resource, and so most texts on the Internet are not likely to be useful for particular educational needs. Thus, finding pedagogically relevant texts can be difficult.
- Second, new texts are typically not accompanied by the sorts of useful instructional content that textbooks provide (e.g., practice exercises, assessments, and lesson plans, etc.).

Regarding the first challenge, tailored text retrieval applications could make it easier for teachers

[2] These two examples are taken from some discussions with teachers about situations in which they use outside texts.
[3] See `http://www.readersworkshop.org/` for details on the Reader's Workshop.

to find texts with specific topics, reading levels, etc. Recent research has explored such applications (Brown and Eskenazi, 2004; Miltsakaki and Troutt, 2008; Heilman et al., 2008).

In this work, we focus on the second challenge—or, at least, a small part of it. By developing a tool to automatically generate questions about new texts, we are facilitating the creation of instructional content such as practice exercises and assessments.

1.3 Types of Questions

In this work, we focus primarily on QG about explicit factual information (as in example 1.4). We now more precisely define the types of questions we aim to generate.

There are many types of questions, and researchers have proposed various taxonomies for organizing them. One particularly useful and concise discussion of the dimensions by which questions can be classified is provided by Graesser et al. (2008). They discuss how questions can be organized by the following characteristics: their purpose, the type of information they seek, their sources of information, the length of the expected answer, and the cognitive processes they involve. This work addresses a small area of that space, as follows.

1.3.1 Purpose

Following Graesser and Person (1994), Graesser et al. (2008) enumerate the purposes of questions: the correction of knowledge deficits (e.g., sincere information-seeking questions such as *How do I get to that restaurant?*), the monitoring of common ground (e.g., a science teacher asking, *Do mammals lay eggs?*), the social coordination of action (e.g., *Would you hand me that piece of paper?*), and the control of conversation and attention (e.g., *How are you doing today?*). This work focuses on monitoring common ground by assessing the knowledge that a student possesses about a particular topic.

1.3.2 Type of Information

Graesser et al. (2008) propose a set of 16 categories for questions based on the type of information involved, ranging from simple to complex questions. These categories were derived from previous

work by Lehnert (1978) and Graesser and Person (1994). We focus on the simple end of the spectrum, with the goal of achieving a system that is scalable to large data sets and new domains. A system for generating more complex questions would be possible, but it would likely require encoding significant human knowledge about the targeted domain and the types of questions.

Specifically, this work aims to generate two types of questions: concept completion questions and, to a lesser extent, verification questions. Concept completion questions elicit a particular information that completes a given partial concept or proposition (e.g., *Who served as the first Secretary of State under George Washington?*). Verification questions invite a yes-no answer that verifies given information (e.g., *Was Thomas Jefferson secretary of state?*). For comparison, other question types include example questions (e.g., *What is an example of an important document written by Thomas Jefferson?*), goal orientation questions (e.g., *Why did Thomas Jefferson support the Embargo Act of 1807?*), and judgmental questions (e.g., *Were Thomas Jefferson's domestic policies successful?*).

1.3.3 Source of Information

This work aims to generate questions for which the source of answer is the literal information in the text—specifically information that is focused at the sentence or paragraph level rather than spread across the whole document. The questions will not address world knowledge, common sense, opinions of the answering party, etc.

1.3.4 Length of the Expected Answer

This work mostly involves questions whose expected answers are short, usually a single word or short phrase. The expected answers are not essays, for example.

1.3.5 Cognitive Processes

The questions generated by this work mainly assess recognition and recall of information—and, to a much lesser extent, comprehension. Very little inference, application, synthesis, or other complex processing is involved. Thus, in terms of Bloom's (1956) taxonomy of education objectives, we are mainly focused at the "knowledge" level. Higher levels of that taxonomy include "comprehension"

(e.g., restating something in one's own words), "application" (e.g., using an equation to solve a practical problem), "analysis" (e.g., recognizing errors), "synthesis" (e.g., writing an essay), and "evaluation" (e.g., selecting the best design for an engineering task).

Clearly, we are focusing on a small part of the space of possible questions. In Chapter 2, we elaborate on why this is still quite a challenging task—and one with connections to existing work in parsing, entity recognition, coreference, and other research areas in computational linguistics. We also discuss some of the challenges of going beyond this small subspace and suggest possible pathways for future QG research.

1.4 Educational Value of Factual Questions

By focusing on explicit factual information, we are restricting the range of useful questions we might be able to produce. Factual questions make up only a small part of the questions used by educators in practice exercises and assessments. For example, educators also ask questions to assess the ability of students to make inferences, as well as their ability to perform various reading strategies such as summarizing or making connections to prior knowledge (National Institute of Child Health and Human Development, 2000, Chapter 4, Part II). However, automatic factual QG may still have the potential to help teachers create instructional content.

Let us first consider the relative benefits of "deeper" inference questions and "shallow" factual questions of the sort we aim to automatically generate. Many studies have explored whether asking higher level, deeper questions leads to better learning than lower level factual questions that focus on recognition and recall. Redfield and Rousseau (1981) and Winne (1979) provide meta-reviews. Most relevant studies focus on primary and secondary education levels. They involve either training teachers to ask high- or low- level questions (and then allowing them to ask what they want to), or explicitly asking teachers to ask certain types of questions. These studies focus primarily on verbal questions asked during class, but are still somewhat relevant to our research on text-specific QG.

While some studies show relatively strong advantages for questions that address higher cognition—several such studies are discussed by Redfield and Rousseau (1981)—many studies find no significant difference between asking shallow and deep questions. In a review of research on question

asking, Samson et al. (1987) found that 88% of the 53 studies they reviewed yielded no significant differences in measures of learning between groups of students that were either asked primarily high-level questions or primarily low-level questions. Some studies even find favorable results for shallow, factual questions (Winne, 1979), and Ryan (1973) suggests that lower level questions may be more useful for lower level students.

Deep and shallow questions may complement each other: lower-level questions ensure that students possess the basic knowledge needed to answer higher-level questions. Higher-level questions ask students to manipulate various pieces of previously acquired information in order to formulate an answer. However, such questions implicitly assume that students have already acquired the basic information to be manipulated by higher level cognition—which may not be the case. As stated by Walsh and Sattes (2004), "students must be able to retrieve information if they are to use it in more cognitively complex operations." Initial lower level questions may be useful for ensuring that students grasp necessary factual knowledge before proceeding to higher level questions. Thus, Willen and Clegg (1986) recommend employing both low- and high-level questions.

Another relevant piece of evidence is that teachers tend to ask many lower-level, shallow questions—in fact, some research indicates that as few as 20% of teacher questions require high-level thinking (Gall, 1984). Even if teachers do ask an overly high proportion of many shallow questions, as argued by Graesser and Black (1985), it seems reasonable to infer from their extensive use that shallow questions have considerable educational value.

Factual questions are also prevalent in written tests and assignments. For example, Ozuru et al. (2008) analyzed questions on the grades 7–9 version of the popular Gates-MacGinitie Reading Test.[4] They found that about a quarter were "text-based" questions, defined as questions that match sentences in the source text nearly verbatim and could be achieved by applying simple transformations—broadly similar to the transformations that we discuss in Chapter 3. They also found that another quarter of the questions involved somewhat more complex transformations such as rewording and syntactic restructuring. A few of these types of transformations, such as the extraction of information from embedded clauses and the resolution of pronouns, are captured by our approach (§3.2 of Chapter 3). Further extensions to our approach could improve its coverage of such transformations. Of

[4] See `http://www.riversidepublishing.com/products/gmrt/` for more details on the Gates-MacGinitie Reading Test.

course, Ozuru et al. (2008) did find that about half of the Gates-MacGinitie questions were bridging or inferential questions—the sorts of questions that are beyond the scope of this work, and perhaps beyond the reach of current NLP techniques. Ozuru et al. (2008) observe that many different levels of questions are needed when working with students of a wide range of ability levels. They also analyzed a version of the test for grades 10–12 and found fewer text-based questions (about 5%) and more bridging and inferential questions (about 75%), suggesting that as grade levels increase, deeper questions become more prevalent.

Asking shallow questions and focusing on factual information may be useful when deeper questions would be too difficult, particularly for struggling students dealing with challenging new material. Research suggests that teachers should pose questions that students have a good chance of answering correctly, in order to avoid discouragement and other negative motivational effects (del Soldato and du Boulay, 1995; Malone and Lepper, 1987), and when students are struggling with the text, simpler questions focusing on explicit factual information may be challenging enough.

An important point is that while human teachers are capable of generating either "deep" questions involving complex inference or "shallow" factual questions, automatic techniques are much more likely to be error prone when complex inference is involved than when it is not. On the other hand, automated QG tools may be capable of generating large sets of shallow questions very quickly and could help teachers to focus on generating good deep questions. Another important point is that exercises and assessments often consist of many types of questions: automatically generated factual questions could be complemented by manually generated deeper questions. Thus, our goal is not to fully automate the process of creating questions, but rather to assist teachers by freeing up their time and reducing their cognitive load. In Chapter 5, we test whether we can achieve this goal; specifically, we develop a QG tool and conduct a user study to test whether teachers can use the tool to create factual questions more efficiently than when they have to create them on their own.

1.5 Comparison to the Cloze Procedure

When automatically generating factual WH questions, there are a variety of challenges, as discussed in Chapter 2. For example, WH questions may be ungrammatical if the automatic parser used in a

QG system provides an incorrect syntactic analysis (§2.2.1). There exists at least one alternative type of reading assessment that is easier to automate and perhaps more effective for certain applications: namely, the cloze procedure (Taylor, 1953).[5]

To generate a cloze test, one takes a passage and replaces some of the words in the passage with blanks. There are various ways of choosing words to replace, the simplest being to choose every Nth word. The reader's task is then to identify the original words that fill in the blanks. Researchers have found that such cloze tests are effective for measuring first language reading comprehension (Bormuth, 1967; Rankin and Culhane, 1969) as well as second language ability (Oller, 1972). Cloze tests can also be automated without introducing many errors (see, e.g., Mostow et al., 2004) since they only require tokenization of an input text into words.[6]

One important difference between cloze questions and the WH questions we focus on is that a cloze question (i.e., a sentence with a blank in it) is usually presented to the student either in the context of the passage from which it came or immediately after the passage. That is, cloze questions for reading assessment are typically "open book" questions. In contrast, we aim to generate WH questions that could be used to assess recall of specific information at some point after the student read the text (i.e., in a "closed book" scenario).[7]

Of course, single sentence cloze questions could be presented outside of a source passage, but then automation becomes more challenging. Taking information out of context can lead to various issues such as unresolved pronouns or other types of reference. For example, consider the following cloze question.

(1.12) He then became the _____ President of the United States.

For a text where multiple U.S. Presidents are mentioned, a question like this would be vague, permitting multiple answers. Thus, for "closed book" applications, cloze questions, like factual WH questions, become challenging to generate automatically. Similar discourse-related issues are a ma-

[5] Cloze items go by various names such as "gap-fill," "fill-in-the-blank," and "sentence completion" questions.
[6] A free online tool for generating cloze questions is provided at `http://www.lextutor.ca/cloze/`.
[7] Factual WH questions could be used for "open book" applications as well, as in the work of Gates (2008).

jor challenge in generating factual WH questions, as discussed in §2.3 of Chapter 2.[8] Context also affects cloze questions for vocabulary assessment (Pino et al., 2008; Skory and Eskenazi, 2010) since a particular target vocabulary word (which would be blanked out) can only be identified by a student if a sufficiently informative context is provided.

A distinct advantage of WH questions over the cloze procedure is the naturalness of WH questions. WH questions are a type of language that people encounter in both spoken and written language, whereas filling in blanks in incomplete sentences is an artificial task. As such, WH questions can be used more readily in a wider range of applications. For example, WH questions could be used in text-based or spoken tutorial dialogue applications, where maintaining natural interactions with users is an important desideratum (Johnson et al., 2000)

Since WH questions and the cloze procedure are in a somewhat different space, and since their utilities may vary considerably depending on the application, we leave direct experimental comparisons to future work.

As a more general point, it seems that research on WH questions is more likely to lead to better understanding of how to automatically generate a wider variety of questions, including deeper ones and those involving other types of structural transformations (e.g., questions involving paraphrasing or textual inference). We discuss the potential for such extensions later in §6.2.2 of Chapter 6.

1.6 Prior Work on Intelligent Tools for Education

A driving motivation for this work is to create tools to help teachers generate instructional content. In this section, we discuss connections to related research on the use of technology and artificial intelligence in education. Later, §1.9 addresses related work on the specific problem of QG.

Many applications of computer technology for assisting teachers involve hardware, such as clicker devices for gathering student responses (Trees and Jackson, 2007) and smartboards (Smith et al., 2005), or standard productivity software such as Microsoft PowerPoint (Szabo and Hastings, 2000). In this work, we focus on creating a tool using techniques from artificial intelligence—specifically,

[8] In Chapter 3, we present an overgenerate-and-rank framework for QG. Such a framework could be adapted for cloze questions or other types of questions. For example, for cloze questions, one could replace the second stage for converting statements into questions with a stage for replacing words with blanks. One could also adapt the features set used in the statistical ranker in stage 3 to address characteristics of cloze questions.

from natural language processing.

The study of artificial intelligence in educational technologies, particularly of intelligent tutoring systems (Koedinger et al., 1997; Vanlehn et al., 2005; Woolf, 2008), is a small but growing field. Many intelligent tutoring systems can be seen as a tool for teachers in that they provide guidance and feedback while students work through practice exercises. In facilitating practice, tutoring systems allow teachers to focus their effort on other issues such as planning curricula and delivering instruction about new concepts. It is worth noting that much of the work on tutoring systems has focused on interactions between an individual student and the computer, and less research on how tutoring systems affect teachers—though there is definitely research on such issues, particularly work by Feng and Heffernan (2006) and Ainsworth (2004).

With respect to intelligent tutoring systems, research on authoring tools for tutoring systems is particularly relevant to this work (e.g., Koedinger et al., 2004; Razzaq et al., 2009). For example, Ritter (1998) describes an authoring tool for automatically parsing the text of an algebra word problem into a formal semantic representation that could be loaded into a cognitive tutor. While their task and techniques differ from what we explore in this work, they have the same high-level goal of generating instructional content from text. A major issue in the research on and development of intelligent tutoring systems—and instructional technology more generally—is the efficiency of content generation and development. For example, it has been estimated that for intelligent tutoring systems, approximately 200 hours of development are required per hour of instruction (Anderson et al., 1995). For tutoring systems with substantial amounts of text content, opportunities may exist for NLP applications like the one we describe to increase the efficiency of system development.

Some of the challenges of authoring content can be addressed by intelligently re-using work by human teachers. For example, Aleahmad et al. (2008) describe a crowd-sourcing approach to the problem of generating content. Focusing on math exercises, they propose creating an online repository of materials by eliciting contributions from teachers and other Internet users. Such content could potentially be used within a tutoring system, or perhaps more directly by classroom teachers. Note that in such an online system for sharing or creating content, automated techniques such as the QG system we describe here could provide "seed" content for contributors to select and revise.

There has also been work on automatically creating content in the area of computer-assisted lan-

guage learning. For example, Meurers et al. (2010) describe a system that takes arbitrary texts as input and, with NLP technologies, highlights specific grammatical constructions and automatically creates grammar practice exercises. Also, Heilman et al. (2008) describe a system that uses natural language processing and text retrieval technologies to help English as a Second Language teachers find pedagogically appropriate reading practice materials (e.g., texts at an appropriate reading level) for intermediate and advanced language learners.

There has been considerable work on applying NLP to educational problems, but most applications deal with the analysis of student responses rather than the generation of instructional content. For example, tutorial dialogue systems (Litman and Silliman, 2004; Graesser et al., 2005; Boyer et al., 2009) use NLP to analyze students' dialogue moves and respond in pedagogically appropriate ways. Automated scoring technologies (Shermis and Burstein, 2003; Mohler and Mihalcea, 2009; Nielsen et al., 2008) grade student responses to essays or short answer questions. Systems for grammatical error detection (Leacock et al., 2010) analyze student writing to find errors involving prepositions, determiners, etc. And, finally, tools for analyzing discussion boards (McLaren et al., 2009) use NLP to provide teachers with efficient ways of monitoring discussions among large groups of students.

This work on automatic QG contributes to the literature on educational applications of artificial intelligence techniques by exploring a problem that combines various aspects of the related work described above. Relatively little past work has focused on NLP or AI tools for automatically generating instructional content and, in particular, how such tools would be used by educators. We explore such an application and, in Chapter 5, present a user study involving potential users (K-12 teachers).

1.7 Prior Work on Overgeneration-and-Ranking

Our approach to QG, described in more detail in Chapter 3, is based on an "overgenerate-and-rank" strategy. The system generates a large set of candidate questions using existing NLP tools and manually written rules. It then ranks candidate questions using a statistical model of question quality.

The general idea of ranking a system's output using statistical methods is nothing new and has been explored in various previous research. For example, Collins (2000) describes methods for re-

ranking syntactic parse trees from a generative parsing model using a discriminative ranker that can consider complex syntactic features. Ranking has also been explored very deeply in the field of information retrieval, particularly for the task of "learning to rank" (LETOR) (Joachims, 2002; Burges et al., 2006; Taylor et al., 2008). Similar to the overgenerate-and-rank approach, a LETOR system takes a set of candidate documents from a simpler retrieval system and re-rank them according to a machine-learned statistical model of relevance.

Overgenerate-and-rank strategies are most closely associated with natural language generation and dialogue systems. Langkilde and Knight (1998) and Langkilde-Geary (2002) discuss methods for efficiently using n-grams statistics gathered from a large general-purpose corpus such as the Penn Treebank (Marcus et al., 1993) to score the outputs from a rule-based generation system in order to improve fluency. Others, such as Varges (2006), extend such work by using domain-specific corpora of human dialogues (e.g., in a dialogue system for choosing restaurants).

Probably the most similar overgenerate-and-rank approach to the one we discuss is that of Walker et al. (2001). Working on a mixed-initiative spoken dialogue system for the travel domain, Walker et al. (2001) use a ranker to select sentence plans, which are formal graph-based representations of potential outputs that their system can generate. They created a tailored dataset to train this ranker by gathering ratings of such sentence plans from trained experts.

In this work, we extend the idea of overgeneration-and-ranking in several ways. First, we apply the technique to a text-to-text generation problem in a broader domain (i.e., informational texts such as encyclopedia articles) than much of the previous work on dialogue systems, which has focused on relatively narrow domains such as travel planning. Second, rather than using a general-purpose corpus (Langkilde and Knight, 1998) or a corpus of expert-generated outputs (Varges, 2006), we create a ranker from a tailored dataset of system outputs rated by humans. And, unlike the system from Walker et al. (2001), which ranks system-internal formal representations, our system ranks natural language outputs (i.e., questions). Third, much of the previous work on ranking focuses on selecting a single top-ranked item (e.g., a syntactic parse or a sentence plan)—with text retrieval (e.g., for web search) being a notable exception. In contrast, we use ranking to generate a *list* of top-ranked questions with the aim of presenting that list to a user. Different questions may address different facts that appear in input texts, and thus having multiple questions to choose from would likely be useful. The

ranker employed in this work serves more to avoid unacceptable questions (and to promote acceptable ones) rather than to select a single output.

1.8 Connections to Other Problems in Natural Language Processing

A number of connections can be drawn between QG and other areas in NLP and computational linguistics.

Our QG scenario can be seen as an instance of monolingual text-to-text generation since both the input and output consist of natural language text. Text-to-text generation has been of great interest to NLP researchers in recent years, with substantial work on problems such sentence compression, paraphrase generation (Callison-Burch, 2007), and text simplification (Beigman Klebanov et al., 2004; Zhu et al., 2010). QG provides several new and interesting challenges in this area, which we discuss in Chapter 2. Connections can also be drawn to question answering, natural language generation, and even machine translation. Table 1.1 summarizes some of the connections.

We speculate that overgenerate-and-rank approaches such as ours may be effective for other text-to-text generation problems. In the overgenerate part of a system, one can encode complex linguistic knowledge in the form of rules. While this ability to encode knowledge may not be as important for problems such as machine translation, where large datasets of aligned input and output sentences are available, it may be very useful for problems such as paraphrase generation where relevant data are scarce. Overgenerate-and-rank approaches also allow one to use machine learning to improve outputs, and our approach of gathering a tailored dataset of human-rated outputs provides a relatively easy way to develop a statistical ranker.

QG also provides motivation for research on core NLP tools (e.g., syntactic parsers, named entity recognizers, and coreference resolvers). As we describe later in Chapter 3, such tools provide useful structural analyses with which input sentences can be converted into questions. Such tools do introduce errors, however, as discussed in §4.11 of Chapter 4, and so it seems that if these tools improved, then QG systems would as well.

NLP Task	Example References	Text In	Text Out	Lex Trans	Syntax Trans	Qstn focus	Comments
Sentence Compression	(Knight and Marcu, 2000; Clarke, 2008)	•	•		•		Compression systems typically produce a single best shortened form of an input sentence, usually by removing words. Syntax plays an important role; lexical analyses are typically shallow.
Lexical/Phrasal Paraphrase Generation	(Callison-Burch, 2007)	•	•	•			Paraphrase generation systems transform input sentences by replacing individual words or phrases with similar ones. Syntactic paraphrasing has been less widely studied.
Text Simplification	(Beigman Klebanov et al., 2004; Zhu et al., 2010)	•	•	•	•		Simplification systems apply both lexical and syntactic transformations to produce simpler versions of declarative sentences.
Machine Translation	(Brown et al., 1990; Och and Ney, 2003)	•	•	•	•		Machine Translation systems apply both lexical and syntactic (or phrasal) transformations from a source language to a (different) target language.
Question Answering	(Voorhees, 2004; Echihabi and Marcu, 2003; Wang et al., 2007)	•		•	•	•	Question answering systems lexically and syntactically analyze a question and a corpus of texts in order to select (rather than generate) sentences that answer the question.
Natural Language Generation	(Reiter and Dale, 1997; G. Angeli and Klein, 2010)		•				Prototypical natural language generation systems take a formal meaning representation as input and produce text as output. They are typically tied to a particular domain (e.g., weather reports).
Question Generation	(Kunichika et al., 2004; Mitkov et al., 2006; Heilman and Smith, 2010b)	•	•	•	•	•	QG systems take texts as input and produce sets of questions as input by performing lexical and syntactic transformations. Here, we focus on broad-domain QG.

Table 1.1: Connections between question generation and other problems in natural language processing. The • symbol marks whether each task takes text as input ("Text In"), produces text as output ("Text Out"), involves significant modeling of lexical transformations ("Lex Trans"), involves significant modeling of syntax transformations ("Syntax Trans"), and/or is focused on the phenomenon of question asking ("Qstn Focus"). Note: We do not use the word "transformation" to indicate that transformations are being *explicitly* represented. Many techniques use other representations, such as alignment.

1.9 Prior Work on Question Generation

In this section, we draw connections to the existing research on automatic QG. We begin by discussing some of the relevant work on automatic generation of non-factual questions and other assessment items (§1.9.1). Then, we describe some broad similarities and differences between our approach and previous approaches to factual QG (§1.9.2), before discussing some specific factual QG systems (§1.9.3). We do not provide an exhaustive discussion of the prior research. For additional information on QG, see a report by Rus and Graesser (2009).[9]

1.9.1 Research on Other Question Generation Problems

Various researchers have explored automatic techniques for generating questions for purposes other than assessing factual knowledge. In this section, we discuss some of the work on such non-factual questions.

Liu et al. (2009) automatically generated questions to support students as they write literature reviews (e.g., *What is the research question formulated by X?*). They automatically extract and classify citations from a student's review, and then use templates to generate questions to provide feedback (e.g., about potentially missing content in the student's work).

Piwek and Stoyanchev (2010) discuss some preliminary work on creating dialogues from monologues by converting statements into questions. They aim to use QG to create more engaging and effective ways of presenting instructional content.

Mostow and Chen (2009) explore the generation of questions about the intentions and mental states of characters in narrative fiction. They use the automatically generated question to scaffold and encourage the generation of questions by beginning readers, since it can be a useful strategy for learning to read (National Institute of Child Health and Human Development, 2000).

A number of papers have explored automatic techniques for generating vocabulary questions. Brown et al. (2005) automatically generate multiple-choice questions about lexical relationships such as synonym and antonymy using WordNet (Miller et al., 1990) as a source of lexical knowledge. Heilman and Eskenazi (2007) apply distributional similarity techniques to extract multiple-

[9] Also, the website `http://www.questiongeneration.org` provides information about meetings and other question generation-related research activities.

choice related word questions from large text corpora. (e.g., *Which set of words are most related in meaning to "reject"? A. pray, forget, remember; B. accept, oppose, approve* . . .) Also, Pino et al. (2008) and Skory and Eskenazi (2010) explore the generation of cloze questions from large text corpora for vocabulary assessment.

1.9.2 Broad Similarities and Differences

As we do here, some past research has focused on generating questions about explicit factual information at the sentence level. Other QG techniques also typically generate questions by applying extraction or transformation patterns to syntactic or shallow semantic representations of the input sentences, which are created as a pre-processing step (Mitkov and Ha, 2003; Gates, 2008; Kunichika et al., 2004). Finally, most QG research has focused on the English language, as we do in this work.

Existing QG research differs, however, in several important ways. First, most prior work has used distinct, context-specific extraction or transformation rules to generate questions—for example, having separate rules to extract *who* questions from main clauses, *where* questions from main clauses, *who* questions from relative clauses, *where* questions from relative clauses, etc. (e.g., Gates 2008; Wyse and Piwek 2009). Also, existing work typically generates questions in a single step, combining the process of transforming declarative sentences into questions with various other transformations (e.g., the extraction of information from relative clauses).

Here, we apply combinations of general rules. Our system breaks QG down into a multi-step process: simplified factual statements are first extracted from complex inputs, and then separately transformed into questions by applying sequences of simple, linguistically-motivated transformations such as subject-auxiliary inversion and WH-movement (as in Figure 1.1). This modular architecture reduces the need for many distinct rules that model similar processes, and makes adding new question types relatively easy (e.g., since the generation of *who* questions follows most of the same steps as the generation of *what* questions).

Also, much of the prior work has not explicitly addressed linguistic phenomena such as WH-movement (Ross, 1967; Chomsky, 1977), semantic entailment, and presupposition (Levinson, 1983, Chapter 4). While we do not ascribe to a particularly linguistic theory—or attempt to generate questions in a way similar to how humans do—we leverage existing knowledge about question-related

phenomena to inform the components of our system (we discuss these phenomena in more detail in Chapter 2).

Perhaps most importantly, previous QG approaches have been comprised almost exclusively of manually written rules (of course, these rules usually operate on linguistic representations that are extracted in pre-processing steps by statistical parsers). Here, however, we include as an integral component of our approach a statistical model of question acceptability that is used to rank candidate outputs. To our knowledge, this is the first work on factual QG to use a machine learning approach to improve question quality.

Most prior work has not been extensively evaluated in broad domain scenarios, instead focusing on narrow domains, such as introductory linguistics (Mitkov and Ha, 2003). Moreover, there is not a clear and agreed upon evaluation methodology for QG. Nor are there alternative systems readily available for comparison. Here, we experiment with Wikipedia articles and other informational texts about a variety of topics in order to gain a better understanding of how QG systems perform in a broad-domain setting.

1.9.3 Specific Approaches to Factual Question Generation

Now we discuss some of the specific approaches to QG that have been explored.

Mitkov and Ha (2003) developed a system that uses rules for creating questions from shallow parses of specific types of sentences (e.g., a rule for creating a question *What do/does/did the* `SUBJECT VERB`? from a sentence with subject-verb-object order). They also describe a technique for generating foils for multiple-choice questions by searching a corpus for noun phrases that are semantically similar to the answer phrase. Mitkov and Ha (2003) evaluated their system by judging the quality of questions generated from an online linguistics textbook. Also, in a study in which two university students checked and revised automatically generated questions, Mitkov et al. (2006) demonstrated that automatic generation and manual correction of questions can be more time efficient than manual authoring alone, which is particularly relevant given that the output of many QG systems would likely require vetting by humans. Chapter 5 describes a user study we conducted involving K-12 teachers that corraborates and extends this finding.

Kunichika et al. (2004) describe a system for generating questions based on syntactic and semantic analyses which are derived using Definite Clause Grammar (Pereira and Warren, 1986). Kunichika et al. (2004) tested their system by judging whether question generated from two English as a Second Language textbooks were "semantically correct," but their exact evaluation criteria and procedure are not very clear.

Gates (2008) describes a QG system aimed at generating questions to present *while* a student reads an article, rather than afterwards, in order to encourage the student to "look-back" and re-read the text (e.g., if he or she does not know an answer). Since such questions are presented together with the source text, Gates sidesteps the important challenges that result from taking questions out of their original discourse context, which we discuss in §2.3 of Chapter 2. Similar to the approach we describe in Chapter 3, Gates's system uses phrase structure parses and the `Tregex` tree searching language.

The overgenerate-and-ranking approach to QG that we describe in Chapter 3 was originally presented by Heilman and Smith (2009) and Heilman and Smith (2010b). The component described in §3.2, which takes complex sentences as input and extracts simplified factual statements that can more readily be converted into questions, was originally described in Heilman and Smith (2010a).[10]

In 2010, there was a shared task evaluation challenge on factual QG (Rus et al., 2010). There were two tracks: one for QG from single sentences (in which four groups participated); and one for QG from paragraphs (in which one group participated).[11] Two of the QG shared task submissions were particularly interesting in that they explored QG from semantic representations, which is a deeper level of linguistic analysis than that which we consider here (since we focus on syntactic and lexical analyses). First, Mannem et al. (2010) presented a QG approach based on semantic role labeling (Carreras and Màrquez, 2005), which identifies and labels the arguments of individual predicates (usually verbs and nominalizations of verbs) with their semantic roles (generalizations of agent, patient, etc.). Second, Yao and Zhang (2010) describe an approach in which questions are generated from sentence-level semantic parses based on minimal recursion semantics (Copestake et al., 2005).

[10]The rule system used for simplified factual statement extraction in stage 1 in (Heilman and Smith, 2010b) and Heilman and Smith (2009) was a prototype version of what we describe in Chapter 3 (§3.2). There are also some minor differences in the rules for question creation in stage 2 (§3.3).

[11]We did not participate in the shared task due to time constraints and the fact that the focus of the task was somewhat different than our research goals, though we were involved with some of the early planning stages.

Such semantic representations may be advantageous for QG in that they provide a more general representation of meaning that abstracts away from particular syntactic constructions to enable simpler, higher-level transformations. It would be interesting to more rigorously compare syntax-based QG systems to semantics-based QG approaches, but we leave such comparisons to future work. We return to the topic of alternative linguistic representations in §6.2.1 of Chapter 6.

A number of interesting ideas have come out of the QG workshops and the shared task on QG, but at the time of writing there still seems to be a lack of consensus in the nascent QG community about what tasks to focus on, what evaluation methodologies to employ, and what linguistic representations are most useful. In this work, we provide a detailed example of the development and evaluation of a QG system, going beyond the mostly preliminary work presented at the workshops. Future work on QG may benefit from our specific descriptions and results, even if alternative approaches to the problem are taken.

1.10 Primary Contributions and Thesis Statement

This section summarizes the main contributions of this research and presents a concise thesis statement.

1.10.1 Summary of Primary Contributions

In this work, we make the following contributions to the literature on natural language processing and educational technologies:

In Chapter 2, we analyze the linguistic and computational challenges involved in automatic factual question generation, which, as we have described, is a problem that has substantial connections to other areas in NLP (§1.8) and has potential for educational applications (§1.4). Also, as discussed in §1.6, work in this area contributes to the literature on artificial intelligence in education and educational applications of NLP.

In Chapter 3, we describe an overgenerate-and-rank approach to QG, which usefully decomposes QG into three steps: the extraction of factual statements from complex input texts, the transformation of factual statements into candidate questions, and the ranking of candidate questions. This frame-

work enables us to leverage existing NLP tools as well as linguistic knowledge about relevant phenomena such as WH-movement. A key feature of this framework is the ranking step, which uses a statistical model of question quality to rank system output so that better question are more likely to be presented first to users. As discussed in §1.7, our specific approach makes novel contributions to the literature on overgeneration-and-rank techniques.

In Chapter 4, we develop intrinsic evaluation methodologies for QG. We evaluate the effectiveness of individual components of our implemented QG system, including the question ranker and the component for extracting factual statements from complex sentences. We also test the performance of the end-to-end system at generating acceptable questions for an entire document and for individual sentences. We find that ranking roughly doubles the acceptability of top-ranked WH questions, but that acceptability is still only around 40–50%. In an analysis of errors made by the system, we observe that a variety of factors affect performance, including poor syntactic parsing and the difficulty of taking factual information out of context.

In Chapter 5, we present the results of a user study involving K-12 teachers that demonstrates the potential for tools based on QG technologies to have real-world educational impact. The user study found that teachers could use a QG tool to generate quiz questions more quickly and with less effort than writing questions on their own. However, the QG tool appeared to have an effect on the types of questions generated, leading to a somewhat higher proportion of simpler questions. We also explore how teachers might use QG tools by studying how often certain features of the interface were used, and by analyzing the sorts of revisions that teachers made to automatically generated questions. To our knowledge, this is the first work on the generation of factual WH questions to study how educators could use a QG tool to create questions about reading materials.

1.10.2 Thesis Statement

Automatic factual question generation can be effectively modeled using an overgeneration and statistical ranking approach. The techniques that we describe, particularly those for simplified factual statement extraction and question ranking, provide measurable advantages over alternative techniques in terms of the quality and yield of the outputs produced by our system. As demonstrated by the results of a user study with grade school teachers, QG tools have the potential to support realistic

educational applications and to make a broader impact on educational technologies.

Chapter 2

Challenges in Question Generation

In this chapter, we present a discussion of the computational and linguistic challenges that arise in automatically generating factual questions for reading assessment. The discussion, and the examples presented herein, should serve the following purposes:

- to illustrate that factual QG is a challenging, many-faceted problem;
- to identify the scope of this work, since we do not address all of the challenges; and
- to draw connections to other problems in computational linguistics and to suggest how techniques and tools for these problems could support automatic QG.

We broadly categorize QG challenges into four categories: lexical challenges (§2.1), syntactic challenges (§2.2), discourse-related challenges (§2.3), and other challenges related to the use of QG tools in classrooms (§2.4).

We use the symbol * to indicate that a sentence is is ungrammatical, and the symbol $\otimes$ to indicate that a sentence is grammatical but unacceptable for other reasons (e.g., vagueness).

2.1 Lexical Challenges

We begin by discussing issues in QG related to lexical semantics (i.e., the meaning of words and short phrases).

2.1.1 Mapping Answers to Question Words and Phrases

The semantics of the answer to a question affect the question's form. In particular, the answer determines whether the question will be a *who* question, a *where* question, etc. Consider taking example 2.1 and generating questions for which *Herman Melville* is the answer. Since Melville was a person, the *who* question 2.2 is acceptable, but the *what* question 2.3 is not.

(2.1) Herman Melville wrote *Moby Dick*.

(2.2) Who wrote *Moby Dick*?

(2.3) $\otimes$ What wrote *Moby Dick*?

Mapping answers to WH words and phrases such as *who* or *which 19th century author* is difficult for (at least) two reasons.

First, while we can construct a fairly reliable map from semantic types to WH words (e.g., persons map to *who*, locations map to *where*—there are more details on this in §3.3.2 of Chapter 3), correctly identifying the high-level semantics requires substantial lexical knowledge. And, while NLP offers potential tools to solve this problem—including named entity recognizers (Finkel et al., 2005) and lexical resources such as WordNet (Miller et al., 1990)—these tools are error-prone.

For example, consider generating a question from example 2.4 where the answer would be *Kongens Nytorv*.[1]

(2.4) Kongens Nytorv, or King's New Square, is the largest open-air square in Copenhagen.

To a person or computer not fluent in Danish, the phrase *Kongens Nytorv* looks very much like a person's name. An automatic QG system might therefore identify it as a person, and produce a *who* question instead of a *what* question, as in question 2.5.[2]

(2.5) $\otimes$ Who is the largest open-air square in Copenhagen?

[1] Many of the example inputs in this section are from the *Encyclopedia Britannica* and Wikipedia datasets described in §4.1.

[2] This entity tagging error is taken from one of our experiments. An interesting side point to consider in relation to this example is that WordNet (Miller et al., 1990) contains a sense for *square* that is a type of person. E.g., *Julie called Bob a square because he stays home on weekends*. We mark question 2.5 as a poor question ($\otimes$) but not an ungrammatical one ($*$) because, like Chomsky's (1957) famous example, *Colorless green ideas sleep furiously*, it is syntactically correct but nonsensical under a normal interpretation.

The second issue is that correctly identifying the semantic type is not always sufficient. Sometimes a simple *what* or *who* question can be vague when the question is taken out of its context. Consider sentence 2.6.

(2.6) A major concern of the Indian government is the trafficking of wildlife products such as tiger and leopard skins ...

While we might correctly identify that *trafficking* is not a person, place, time, etc., generating a *what* question such as 2.7 is unacceptable since the answer to such a question is not well specified (i.e., there are probably many major concerns of the Indian government).

(2.7) $\otimes$ What is a major concern of the Indian government?

Question 2.8 is a more specific and acceptable question.

(2.8) What illegal activity is a major concern of the Indian government?

However, mapping the answer phrase *trafficking of wildlife products ...* to *illegal activity* (or something even more specific such as *illegal activity involving animals*) would require very detailed and context-specific lexical knowledge of the word *trafficking*—something that current lexical resources and models of lexical semantics do not provide with high accuracy.

This phenomenon is, of course, not solely lexical—in particular, the discourse context will determine whether questions such as example 2.7 are sufficiently specific. We discuss similar discourse-related issues in §2.3.

In this work, we address the first issue of deciding the semantic types of answers and mapping them to WH words (§3.3.2 of Chapter 3). However, we leave to future work the more difficult problem of generating more complex question phrases such as *what illegal activity* based on context.

2.1.2 Variation and paraphrasing

Another challenge in QG is that, ideally, questions should exhibit some variation from the input text. Also, one might want to generate a variety of wordings for a particular question. For example, one might want to generate all of the following questions about Herman Melville, which are roughly equivalent in meaning but vary in their surface realizations:

(2.9) Who wrote *Moby Dick*?

(2.10) Who authored *Moby Dick*?

(2.11) Who was *Moby Dick* written by?

(2.12) Who was *Moby Dick*'s author?

Of course, this is not just a lexical issue—syntactic variation is also important (e.g., converting between passive and active voices, nominalization of verbs, etc.).

One possible way to achieve such variation would be to paraphrase input sentences in an initial step of an algorithm, and then proceed to convert the paraphrased versions into questions.[3] Existing paraphrase generation techniques (Callison-Burch, 2007; Kok and Brockett, 2010) could be adapted for such an application, but these techniques are still relatively new and error-prone. For example, in an intrinsic evaluation of paraphrase quality, Callison-Burch (2007, p. 75) found that only 57% of automatically generated paraphrases were grammatical and retained the meaning of the original input. Since the cost of frequently introducing bad questions with paraphrase errors seems to outweigh the benefit of increased variation, we leave this issue to future work.

2.1.3 Non-compositionality

The final lexical issue we discuss is that the meaning of phrases and sentences is not always well modeled as a compositional process—that is, the meaning of a phrase is not just some simple aggregation of the meaning of its component words. Non-compositionality is an issue because the sorts of representations that are useful in QG, and in NLP more generally, often make assumptions of compositionality. For example, the syntactic representation produced by most parsers (which we make use of in the approach described in Chapter 3) represents each word token as independent of its syntactic siblings given its parent.[4] This makes it difficult to model multi-word expressions such as *burned to the ground* in example 2.13.

(2.13) When Russia invaded Finland in 1808, Helsinki was again burned to the ground.

[3] Such transformations could be included in the first stage of our framework for QG (Chapter 3).

[4] It is worth noting that the Penn Treebank (Marcus et al., 1993), the most widely used annotated resource for training English parsers, does not annotate multi-word expressions.

The phrase *burned to the ground* is essentially a single linguistic unit—sometimes called a construction (Goldberg, 2006)—that cannot be broken up into parts. However, a QG system using transformations such as those in Figure 1.1 (page 3) might identify *the ground* as a potential answer, and produce the unacceptable and awkward question 2.14.

(2.14) ⊗ What was Helsinki again burned to?

Clear idioms such as *burned to the ground* are not the only sorts of multi-word expressions that present a challenge for automatic QG. Also problematic are verb-object combinations such as *take control* (e.g., ⊗ *What did John take?*) or *make changes* (e.g., ⊗ *What did the manager make?*), and other strong collocations such as *chaos ensued* (e.g., ⊗ *What ensued?*).

Techniques for modeling multi-word expressions are still an active area of research (Sag et al., 2002; Katz and Giesbrecht, 2006; Fazly et al., 2009), so we leave this problem to future work. That is, the QG system we develop does not include a specific component to avoid or rephrase questions involving multi-word expressions, and as such, may produce errorful questions when they appear (§4.11 of Chapter 4 discusses an error analysis study that provides a rough estimate of how often such errors occur).

2.1.4 Answers and Distractors

Another issue is that factual questions have answers associated with them, and there are often multiple possible realizations of the correct answer for a given question. For example, in response to the question, *Who wrote* Moby Dick*?*, one might respond with any of the following: *Herman Melville*, *Melville*, *H. Melville*, or *the 19th century American author Herman Melville*. On the other hand, there are often numerous responses that exhibit varying degrees of incorrectness (e.g., *an American author*, *Herman*, *Hermon Melvel*, or *Nathaniel Hawthorne*).

Users of QG tools would likely want to be able to automatically evaluate responses to automatically generated questions. There are two main options for automated grading:

One could attempt to create multiple-choice items by generating distractors (i.e., plausible incorrect answers) that could accompany automatically generated questions. Mitkov and Ha (2003) take this approach and describe a method for distractor generation. Alternatively, one could allow stu-

dents to respond with natural language and apply techniques for automatically scoring short answers (Sukkarieh and Bolge, 2008; Nielsen et al., 2008; Bailey and Meurers, 2008; Mohler and Mihalcea, 2009), many of which build on research on related problems such as recognizing textual entailment and paraphrase identification (Dagan et al., 2005; Bar-Haim et al., 2007; Das and Smith, 2009; MacCartney, 2009; Heilman and Smith, 2010d). Since both of these approaches are somewhat orthogonal to the process of generating questions, we leave the issue of evaluating answers to future work.

2.2 Syntactic Challenges

This section discusses some of the major challenges in QG that are related to the syntactic structure of natural language.

2.2.1 Shortcomings of NLP Tools for Analyzing Syntactic Structures

To generate questions, QG systems need to model some of the complexities of natural language syntax: for example, they need to do things like identify phrasal boundaries (e.g., which words are part of an answer), identify predicates and their arguments, and extract information from nested structures such as relative clauses.

Here, to model syntax, we use standard tools for phrase structure parsing as a pre-processing step in our QG system (§3.1 of Chapter 3), and thus build off of widely used syntactic representations and existing work on automatic parsing (Collins, 1999; Charniak, 2000; Klein and Manning, 2003).

Unfortunately, current parsers are far from perfect. For example, Klein and Manning (2003) report 86.7% labeled constituent F_1 score when comparing predicted parse trees to gold-standard trees, and 91.0% accuracy for syntactic dependencies, using Wall Street Journal texts from the Penn Treebank(Marcus et al., 1993).[5] Out-of-domain parsing performance is typically lower (Foster, 2010).

When parsers make mistakes, so might QG systems. Consider example 2.15.

(2.15) The city's chief manufactures are processed food, lumber, and textiles.

[5]For details on parsing evaluation metrics, see Klein and Manning (2003). For our purposes, the parsing results in the literature tell us to expect that about 10% of the decisions made by a parser are going to be incorrect.

The Stanford parser[6] incorrectly tags *manufactures* as a verb and produces a tree in which that verb is the main predicate, as shown in Figure 2.1. This incorrect parse is somewhat similar to the correct parse one might expect for the sentence *The city's chief is inspecting processed food, lumber, and textiles*.

Any QG system relying on the incorrect parse in Figure 2.1 is very likely to produce ungrammatical questions in which *manufactures* is the main verb, such as examples 2.16 and 2.17.

(2.16) * Who manufactures are processed food, lumber, and textiles?

(2.17) * What does the city's chief manufacture are?

In contrast, QG systems using a correct parse (also shown in Figure 2.1) are much more likely to lead to grammatical questions such as *What are the city's chief manufactures?*.[7]

It is worth noting, however, that a syntactic parse of an input sentence need not be perfect to generate good questions from it. For example, mis-parsing the noun phrase *the 16th president* in example 2.18 is unlikely to affect the quality of questions that do not involve that appositive, such as example 2.19.

(2.18) Lincoln, the 16th president, delivered the Emancipation Proclamation.

(2.19) Who delivered the Emancipation Proclamation?

In this work, we rely on a parser for syntactic analyses and study how often incorrect parses lead to unacceptable questions. However, we do not study parsing directly since it is a large area of active research in NLP, and since there are other issues more closely related to QG.

[6] We use the `2008-10-26` version of the Stanford Parser (Klein and Manning, 2003) with the lexicalized version of the default grammar (`englishFactored.ser.gz`).

[7] Note how the noun phrase *the city* in this question is vague. We discuss this challenge in §2.3.1.

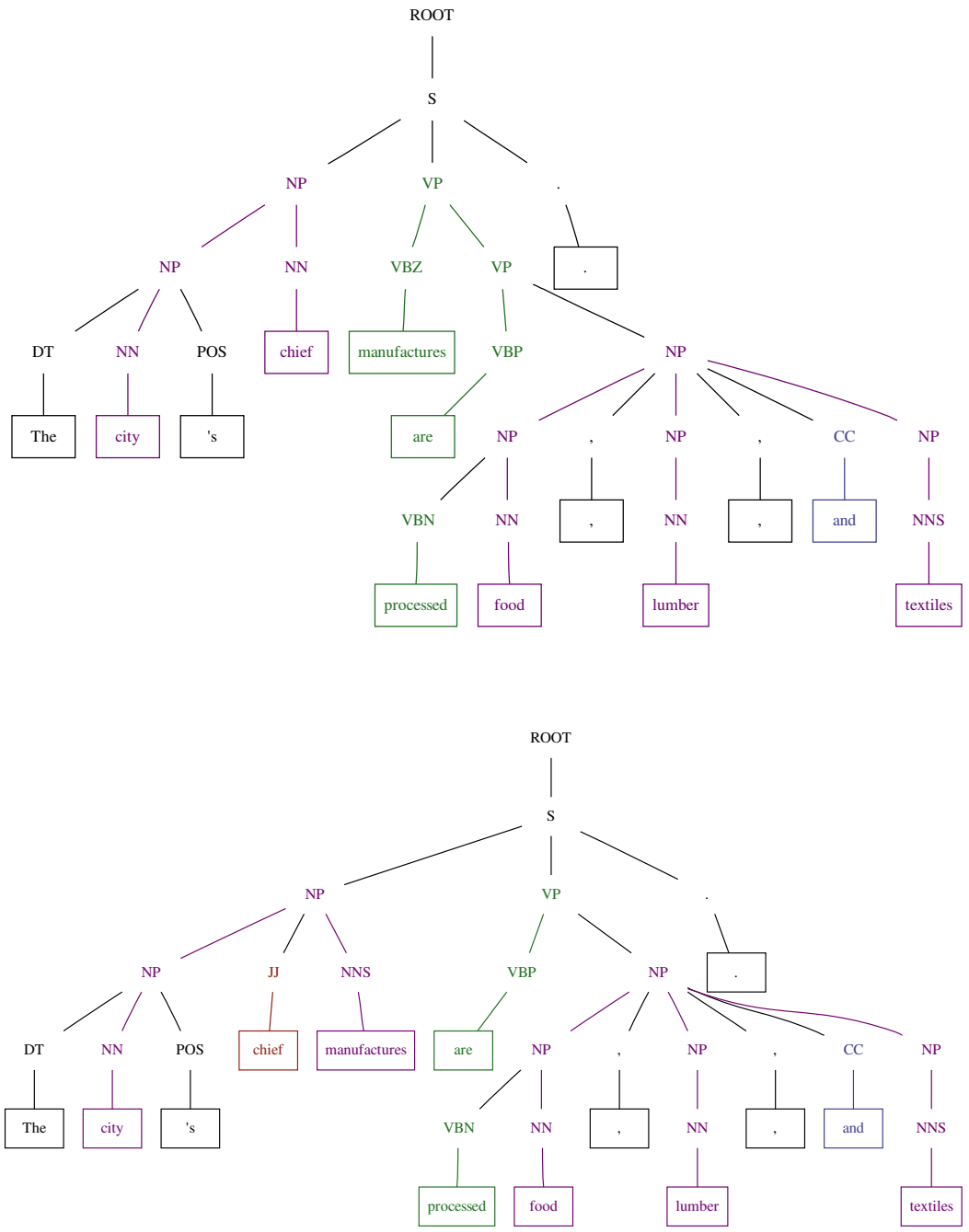

Figure 2.1: Top: An incorrect syntactic parse from the Stanford Parser of the sentence *The city's chief manufactures are processed food, lumber, and textiles*. Bottom: A manually corrected parse of the same sentence.

2.2.2 Variety of Syntactic Constructions

Even if a parser is able to provide an accurate representation of the syntactic structure of a sentence, extracting information from or transforming that structure is nontrivial. In this section, we discuss how complex sentences use various syntactic constructions to present multiple individual facts, each of which we might want to ask questions about.

Factual questions about reading materials are usually short and target a single piece of information. For example, in the Microsoft Research Question-Answering Corpus,[8] which consists of questions generated by students and excerpts from encyclopedia articles that answer them, the average length of questions is 7.2 words. However, sentences in informational texts are typically much longer on average because various constructions such as conjunctions, subordinate clauses, and appositives enable writers to convey multiple pieces of information in a single sentence. For example, for experiments discussed in §4.2 of Chapter 4, we estimate that sentences in Encyclopedia Britannica texts are approximately 23.5 words long. Consider example 2.20.[9]

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

An important issue in QG is how to generate concise questions from complex sentences that appear in passages such as this one, which contains a non-restrictive appositive (*the country's paramount leader*), a participial phrase (*returning ...*), and two clauses conjoined by *so*.

One possible solution is to use techniques from sentence compression (Knight and Marcu, 2000; Dorr and Zajic, 2003). The goal of compression is to take as input a possibly long and complex sentence, and produce as output a single shortened version that conveys the main piece of information in the input. However, as noted above, sentences often convey multiple pieces of information. In QG, we may want to generate questions not just about the information in the main clause, but also about the information embedded in various nested constructions. For example, from the two sentences in

[8] Available at `http://research.microsoft.com`.

[9] Example 2.20 is taken from "Subway Blasts Kill Dozens in Moscow" by C. Levy, *New York Times*, downloaded March 29, 2010.

example 2.20, we might produce the following sentences to allow a QG system to generate a more comprehensive set of questions:

(2.21) Prime Minister Vladimir V. Putin is the country's paramount leader.

(2.22) Prime Minister Vladimir V. Putin cut short a trip to Siberia.

(2.23) Prime Minister Vladimir V. Putin returned to Moscow to oversee the federal response.

(2.24) Mr. Putin built his reputation in part on his success at suppressing terrorism.

(2.25) The attacks could be considered a challenge to his stature.

The task of extracting simple sentences from a complex input sentence is essentially the task of generating a particular subset of the possible sentences that a reader would assume to be true after reading the input. That is, we aim to generate a restricted set of entailed sentences, using the informal definition of entailment from the recognizing textual entailment task (Giampiccolo et al., 2007; Bar-Haim et al., 2007; MacCartney, 2009). While it is important to generate outputs that are true given an input sentence, we are not satisfied with outputs that merely preserve meaning. Rather, we seek short sentences such as examples 2.21–2.25 that are likely to lead to concise questions. §3.2 of Chapter 3 describes our approach for extracting such statements, which is based on knowledge about semantic entailment and presupposition, two phenomena that affect the meanings of embedded clauses in complex sentences.

2.2.3 Constraints on WH-Movement

In other text-to-text generation problems such as sentence compression or machine translation, generating a single acceptable output is typically sufficient. However, in the case of QG, where there are often multiple potential answer phrases in a sentence for an individual input sentence, there are often multiple distinct outputs—that is, outputs that are not simply paraphrases of each other but target different answers—that a system should aim to generate.

However, one cannot simply select each noun phrase in a sentence and formulate a question about it by performing a few transformations. There are complex linguistic constraints, sometimes

referred to as island constraints, that affect which phrases in a sentence can undergo WH-movement, and ignoring these constraints can lead to ungrammatical questions. Consider sentence 2.26.

(2.26) John thought Mary said that James wanted Susan to like Peter.

From this input, we can generate acceptable questions such as 2.27–2.30, but question 2.31 is ungrammatical.

(2.27) Who thought Mary said that James wanted Susan to like Peter? (answer: John)

(2.28) Who did John think said that James wanted Susan to like Peter? (answer: Mary)

(2.29) Who did John think Mary said that James wanted to like Peter? (answer: Susan)

(2.30) Who did John think Mary said that James wanted Susan to like? (answer: Peter)

(2.31) * Who did John think Mary said that wanted Susan to like Peter? (answer: James)

The relevant constraint in this example is that subjects of embedded clauses with overt complementizers (e.g., in this case, the word *that*) typically cannot undergo WH-movement (Chomsky and Lasnik, 1977; Sobin, 1987).[10] This is sometimes called the complementizer-trace effect, or the *that*-trace effect. One could, of course, simply remove *that* in this case to generate the following:

(2.32) Who did John think Mary said wanted Susan to like Peter? (answer: James)

However, such transformations are not always simple, and sometimes they would involve lexically or syntactically paraphrasing the input first. Consider example 2.33.

(2.33) Darwin studied how species evolve.

In order to generate a question with the answer *species* that is grammatical, unlike question 2.34, we would need to transform the phrase *how species evolve*, as in question 2.35.

(2.34) * What did Darwin study how evolve?

[10] Speakers of certain dialects of English may find examples such as 2.31 to be grammatical (Sobin, 1987). Our system conservatively attempts to avoid generating such questions.

(2.35) What did Darwin study the evolution of?

Such paraphrasing is quite difficult with current tools, as we discussed in §2.1.2. Therefore, in this work, we model WH-movement constraints in order to avoid generating ungrammatical questions such as question 2.34, but leave the complex transformations needed to generate questions like question 2.35 to future work.

An important point is that there is no large corpus with questions that violate WH constraints, which could be used as negative examples for training a statistical model of these phenomena. Native speakers rarely, if ever, violate these constraints. One might try using a generative, unsupervised approach to learning these constraints from only positive inputs (i.e., good questions), but we leave that task to future work.

On the other hand, there is considerable linguistic knowledge available about the formation of questions. In fact, the study of questions and WH-movement led to some of the seminal work in linguistics on syntax (Ross, 1967; Chomsky, 1977). While some of the theoretical work may not bear on application development, the linguistics literature provides very detailed descriptions of these phenomena.

Later, in §3.3.1 of Chapter 3, we discuss how to encode some linguistic knowledge about WH constraints to distinguish potential answer phrases that can undergo WH-movement from those that cannot.

2.3 Discourse Challenges

This section describes some of the challenges of QG that involve higher level discourse and pragmatic phenomena. We organize them into two categories: problems with vagueness that result from taking information out of context, and problems that result from implicit discourse relations and computers' lack of world knowledge.

2.3.1 Vagueness of Information Taken Out of Context

Texts are typically written as complete discourses that stand on their own. When new entities or concepts are introduced, writers typically use very specific descriptions such as *Abraham Lincoln*—at

least in the sorts of informational texts we are focusing on. However, individual sentences build on previous sentences, and writers make use of linguistic devices such as anaphora to avoid the repetition of information (Levinson, 1983, Chapter 3). As a result, since an individual sentence or clause may depend on its discourse context, taking facts out of their contexts and generating questions about them can lead to vague and awkward questions.

Pronoun Anaphora

Pronoun anaphors are one very clear example of how questions can be vague out of context. Consider the second sentence in example 2.36.

(2.36) Abraham Lincoln was the 16th president. He was assassinated by John Wilkes Booth.

From this, we would like to generate questions such as 2.37. However, with just basic syntactic transformations, we are going to produce questions such as example 2.38.

(2.37) Who was Abraham Lincoln assassinated by?

(2.38) $\otimes$ Who was he assassinated by?

Question 2.38 could be valid in certain situations, such as if it were asked immediately after a reader read sentence 2.36 (or if a reader knew with certainty that *he* referred to Abraham Lincoln). However, most of the time, such a question is not sufficiently specific to be acceptable, since there are many possible answers (e.g., *Lee Harvey Oswald*, *Mark David Chapman*).

One option is simply to not include any questions that contain pronouns in the output of a QG system. This option seems somewhat unsatisfying since pronouns are quite common. In preliminary experiments, we estimated that pronouns appear in approximately one third of sentences in texts from Wikipedia and the Wall Street Journal. Note that if one is not concerned with identifying answers, then one can generate valid questions where a pronoun would be the correct answer, such as example 2.39—that is, if one adopted the strategy of skipping questions with pronouns, one need not completely skip *sentences* with pronouns.

(2.39) Who was assassinated by John Wilkes Booth?

Another option is to use automatic anaphora resolution techniques to identify the antecedents of pronouns, and then replace those pronouns with their more specific antecedent phrases. We explore this option in §3.2.2 of Chapter 3.

Vague Noun Phrases

Pronouns are, of course, not the only type of anaphora. Noun phrases can also refer to previous information in the discourse, leading to potentially vague questions.

Consider example 2.40, from the Wikipedia article about William Hanna, a cartoon producer.

(2.40) ...The show boosted the studio to the top of the TV cartoon field ....

A straightforward transformation of this sentence would lead to the grammatical but still unacceptable question 2.41.

(2.41) ⊗ What boosted the studio to the top of the TV cartoon field?

It may not be clear to the reader of this question what the phrase *the studio* refers to. A better question would be example 2.42, which uses the more specific phrase *the Metro-Goldwyn-Mayer cartoon studio*.[11]

(2.42) What boosted the Metro-Goldwyn-Mayer cartoon studio to the top of the TV cartoon field?

Avoiding vague noun phrases like *the studio* is somewhat more difficult that avoiding personal pronouns. First, there is not a simple list of words to avoid, as with pronouns. Second, general noun phrase coreference resolution is also a more challenging task than pronoun resolution.[12]

There are certain characteristics of noun phrases that are somewhat, but not completely, indicative of whether they would lead to vague questions (e.g., the type of determiner used, whether the noun phrase includes proper nouns, whether it includes relative clauses, etc.). However, these characteristics are fairly complex and interact with one another. For example, it is not always the case that a question with a proper noun is better than one with a common noun, or that proper nouns are

[11] In this example from Wikipedia, while the answer, *the Flintstones*, appeared in the preceding sentence, the antecedent for *the studio* appeared three paragraphs prior and would be difficult even for a human to resolve.

[12] Haghighi and Klein (2009, Table 3) report a 28% error rate for finding the antecedents of pronouns versus a 35% error rate for nominal mentions that are not proper nouns.

less vague than noun phrases with relative clauses. Therefore, rather than encoding such characteristics as rules, we include them as features in a statistical model of question acceptability that we use to rank questions, as discussed in §3.4 of Chapter 3.

We leave the possibility of replacing vague noun phrases with their antecedents to future work.

Vagueness with Respect to Time, Location, and Situation

Sometimes it is not the presence of context-dependent words in sentences that leads to vagueness, but rather the absence of contextual information. In particular, information about temporal and locational context is often assumed after being introduced early on in a discourse. Consider example 2.43.

(2.43) Typhoon Paka first impacted the Marshall Islands .... Later, it passed just north of Guam, where strong winds destroyed about 1,500 buildings and damaged 10,000 more; 5,000 people were left homeless, and the island experienced a complete power outage following the typhoon.

The second clause (*5,000 people were left homeless* ...) assumes the context of the preceding sentences, which is problematic if we wish to generate a question that may be presented outside of this context. For example, a question such as example 2.44 is likely to be vague because it does not specify the temporal context or the location of the event. Ideally, we would like to incorporate the contextual information into the question, as in example 2.45.

(2.44) ⊗ How many people were left homeless?

(2.45) How many people were left homeless when Typhoon Paka passed just north of Guam?

The statistical model for question ranking that is described in §3.4 of Chapter 3 partially addresses this challenge by helping our system to avoid some of these vague questions. However, we leave to future work more comprehensive solutions that incorporate contextual information from previous clauses and sentences (e.g., as in example 2.45).

2.3.2 Implicit Discourse Relations and World Knowledge

One of the most difficult challenges in generating questions is that writers make extensive use of implicit discourse relations, relying on the world knowledge and inference abilities of readers to make sense of them. Unfortunately, current NLP technologies lack both world knowledge and inference abilities, making it practically impossible to generate many useful questions.

Consider the following example: let us say that we are given the Wikipedia article about U.S. President Abraham Lincoln.[13] One of the key facts about Lincoln is that he was killed by John Wilkes Booth, and so an ideal QG system would include something like the following in its output:

(2.46) Who killed Abraham Lincoln?

Unfortunately, while there is an entire section in the Wikipedia article that discusses Lincoln's assassination, it never explicitly states that Booth killed Lincoln (see Figure 2.2). The closest thing to an explicit statement is (arguably) *Booth jumped into the box and aimed a single-shot, round-slug .44 caliber Henry Deringer at his head, firing at point-blank range*, which would require resolving *his* to Lincoln, recognizing that a *Henry Deringer* is a gun, recognizing that guns fired at point-blank range typically kill the owners of the heads they are aimed at (and do so only when fired), etc.

This lack of explicit information is quite common in natural language: writers and speakers often assume considerable world knowledge and inference abilities (Levinson, 1983). And, of course, NLP tools have neither world knowledge nor the ability to perform complex inference.

Thus, automatic QG about implicit factual information would be much more difficult (at least without the introduction of considerable domain-specific knowledge). Also, the NLP tools needed to go beyond factual questions are mostly non-existent or in fairly early stages of development. For example, while systems for recognizing textual entailment (Dagan et al., 2005) or paraphrase identification (Dolan and Brockett, 2005; Das and Smith, 2009) might help us to recognize some of the implied information to generate questions about, the performance of these system is quite poor in general (e.g., most RTE systems are about 55–65% accurate).

It would probably be even more difficult to generate specific questions that go beyond factual

[13] This example involves the version of the Abraham Lincoln Wikipedia article from August 21, 2008, which can be viewed at `http://en.wikipedia.org/w/index.php?title=Abraham_Lincoln&oldid=233263910`.

Text	Comments
Originally, **John Wilkes Booth**, a well-known actor and a Confederate spy from Maryland, **had formulated a plan to kidnap Lincoln** in exchange for the release of Confederate prisoners.	Booth had plans to do something other than kill Lincoln.
After attending an April 11 speech in which Lincoln promoted voting rights for blacks, an incensed **Booth changed his plans and determined to assassinate** the president.	Booth changed his plans to assassinate the president, but it is difficult to infer that he succeeded. The phrase *the president* might or might not refer to Lincoln.
Learning that the President and First Lady would be attending Ford's Theatre, he laid his plans, assigning his co-conspirators **to assassinate Vice President Andrew Johnson and Secretary of State William H. Seward** . . .	The text indicates that people other than Lincoln may have been assassinated, and persons other than Booth might have been the assassins.
As a lone bodyguard wandered, and Lincoln sat in his state box (Box 7) in the balcony, Booth crept up behind the President and waited . . . . Booth jumped into the box and **aimed a single-shot, round-slug .44 caliber Henry Deringer at his head, firing at point-blank range.**	Henry Deringer looks appears to be the name of a person. Perhaps Henry Deringer killed Lincoln. Also, the pronoun "his" could refer to Booth or Lincoln, so Booth might have been aiming at his own head. Finally, it is not clear how firing at point-blank range will affect the aforementioned person's head.
Major Henry Rathbone momentarily grappled with Booth but **was cut by Booth's knife**.	A person other than Lincoln was harmed by Booth's knife.
Booth then leapt to the stage and shouted "Sic semper tyrannis!" . . . and escaped, despite a broken leg suffered in the leap. A twelve-day manhunt ensued, in which Booth was chased by Federal agents (under the direction of Secretary of War Edwin M. Stanton). He was eventually cornered in a Virginia barn house and shot, **dying of his wounds soon after**.	Someone was shot by Federal agents and died. It may have been Booth, Lincoln, or someone else.
An army surgeon, Doctor Charles Leale, **initially assessed Lincoln's wound as mortal**.	It is not obvious to a computer that "mortal" implies that the wounded person died. The modifier "initially" might imply that the wounded person actually survived.
The President was taken across the street from the theater to the Petersen House, where **he lay in a coma for nine hours before dying**.	The noun phrases *he* and *the President* might or might not refer to Lincoln.

Figure 2.2: The section about Abraham Lincoln's assassination from the Wikipedia article about Lincoln. This excerpt demonstrates that in general, question generation can be extremely challenging. In this case, generating *Who shot Lincoln?* requires a deep, human-level understanding of the text. We focus on more explicit information that computational methods can handle, with the hope of informing future work on deeper questions.

information and ask about an author's opinions, the conclusions that might be drawn from a text, connections with a student's to prior knowledge, etc.

Reliably generating *why* questions is also challenging. Consider example 2.47.

(2.47) As a lone bodyguard wandered, and Lincoln sat in his state box (Box 7) in the balcony, Booth crept up behind the President and waited . . . Booth jumped into the box and aimed a singleshot, round-slug .44 caliber Henry Deringer at his head, firing at point-blank range.

A number of causal relations are present in this example, but none are stated explicitly (e.g., with the word *because*)—and computational approaches for handling implicit discourse relations are still in the early stages of research (Prasad et al., 2008).

One option for getting around the issue of implicit causal relations is to generate *why* questions without knowing the exact answer. However, this can lead to undesirable questions that are obvious just from reading the question. For example, the answer to question 2.48 would be something like *Booth aimed the gun at Lincoln's head because he wanted to kill him*.

(2.48) ⊗ Why did Booth aim a singleshot, round-slug .44 caliber Henry Deringer at Lincoln's head?

Generating *why* questions without knowing the answer could also result in questions whose answers are not in the text at all. For example, the answer to question 2.49 (i.e., the reason for the bodyguard wandering) is never mentioned in the text from which example 2.47 was taken.

(2.49) ⊗ Why did the lone bodyguard wander?

Unfortunately, generating meaningful *why* questions for which we can be sure there is a specific causal relationship in the text often requires making inferences about implicit discourse relations. For example, generating *Why did Booth kill Lincoln?* would involve identifying the possible causes mentioned in the text, but these are mentioned only in indirect ways (e.g., that Booth was a Confederate spy, that Booth wanted to obtain the release of Confederate prisoners, that Booth attended a speech in which Lincoln promoted voting rights for blacks). Prasad and Joshi (2008) provide some further discussion of the challenges of automatically generating *why* questions.

In this work, we focus on facts that are explicitly mentioned in the text, and we do not attempt to generate questions that require world knowledge or complex inference, including *why* questions.

2.4 Challenges Related to Use of Question Generation Tools

In this section, we discuss some of the challenges related to the use of QG tools in realistic settings. To varying extents, these challenges are related to the linguistic ones we have already mentioned, but they also involve human-computer interaction issues.

2.4.1 Importance and Relevance of Information

Not all questions have the same value. Also, the value of a question depends on the particular context in which it will be used. For example, a teacher whose students are reading a text about the city of Dublin might either choose to focus on basic factual information about the city (e.g., *What is Dublin the capital of?*), very detailed facts (e.g., *In what year did Vikings establish the city of Dublin?*), or various other types of information. A teacher might also want to focus on particular aspects of a text because they relate to other topics in his or her curriculum.

A possible approach to estimating the relevance and importance of information might be to adapt NLP techniques for extractive text summarization (Erkan and Radev, 2004; Nenkova, 2006; Toutanova et al., 2007). Algorithms such as maximum marginal relevance (Carbonell and Goldstein, 1998) could also be applied to avoid redundancy in a set of questions. However, since the importance of a question can depend to a large extent on the particular setting in which the question is to be used, we leave to future work the issue of how to automatically decide whether a question is important. One option is to leave the decision entirely up to the user, as we do in the user study described in Chapter 5.

2.4.2 Usability and Human-Computer Interaction Issues

Due to the complexities of language and the limitations of NLP technologies described above, QG tools will make mistakes. In educational applications, it is very important that ungrammatical questions and other types of unacceptable questions not be presented to students. Thus, in general, ed-

ucators need to be part of the QG process, and human-computer interaction issues are important to consider. QG tools need simple, intuitive user interfaces with which educators can select, revise, and possibly share the candidate questions produced by a QG system.

Though these human-computer interaction issues are not a major focus of this work, we return to them in Chapter 5, where we develop and evaluate a prototype QG tool.

2.5 Summary

In this chapter, we described some of the salient challenges in factual QG. We organized these challenges into broad categories: lexical challenges, syntactic challenges, discourse challenges, and other challenges related to the use of QG tools. Next, we present a QG system that addresses many but not all of these challenges.

Chapter 3

Overgenerate-and-Rank Framework for Question Generation

Now that we have introduced the problem of factual question generation and described the challenges associated with it, we present our solutions. This chapter defines a three-stage framework for factual QG question generation, as illustrated in Figure 3.1, and then describes our implementation of that framework.[1] Note that the next chapter, Chapter 4, will intrinsically evaluate various components of the system and compare it to other possible approaches.

In stage 1 of our system, a sentence or a set of sentences from the text is transformed into a simpler declarative statement by (optionally) altering or transforming lexical items, syntactic structure, and semantics. Our implementation includes operations for extraction and simplifying complex sentences and for resolving pronouns. Future work could adapt other NLP transformations such as sentence compression, sentence fusion, or paraphrase generation, for use in this stage.

In stage 2, the declarative sentence is turned into a set of questions by executing a series of well-defined syntactic transformations (WH-movement, subject-auxiliary inversion, etc.).

Since different sentences from the input text, as well as different transformations of those sentences, may be more or less likely to lead to high-quality questions, in stage 3 the questions are scored and ranked according to features of the source sentences, input sentences, question, and trans-

[1] As discussed in §1.9.3 of Chapter 1, much of the content of this chapter was presented in previous publications (Heilman and Smith, 2009, 2010b,a).

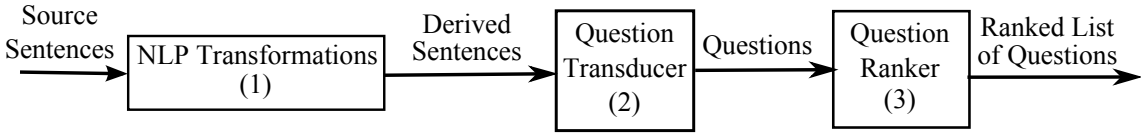

Figure 3.1: Three-stage framework for automatic question generation.

formations used in generation.

A step-by-step example of this process, using a sentence from the Wikipedia article on the history of the city of Los Angeles, is illustrated by Figure 3.2.

3.1 Leveraging Existing NLP Tools

Before presenting the details of how our system generates questions, we now describe some core NLP tools that we employ in our system to analyze the linguistic properties of input sentences.

3.1.1 Stanford Phrase Structure Parser

As a preprocessing step, we use the `2008-10-26` version of the Stanford Parser (Klein and Manning, 2003) to automatically sentence-split, tokenize, and parse input texts—resulting in Penn Treebank-style (Marcus et al., 1993) phrase structure syntax trees for each sentence. For readers not familiar with this representation, Figure 2.1 in Chapter 2 provides examples of accurate and inaccurate parse trees.[2]

We use the lexicalized version of the default grammar (`englishFactored.ser.gz`) that is provided with the parser. We also select noun phrase heads using Collins's head rules (Collins, 1999), which are implemented in the Stanford Parser's API.

For computational reasons, and because parsing accuracy is typically quite low for long sentences, we only parse and generate questions from sentences that have 50 or fewer word tokens in

[2]For sentences with adverbs that immediately preceed the main verb phrase node (e.g., *Tim rarely studied*), the Stanford Parser often returns parses with the adverb phrase as a sister of the main verb phrase. To avoid bad questions such as *Rarely who studied?* and to keep later transformations from having to deal with this issue, we move any such adverb phrase nodes ("`ADVP`") to be the first children of the main verb phrase node ("`VP`"). Note that this does not change the actual word ordering in the sentence, only the structure of the tree.

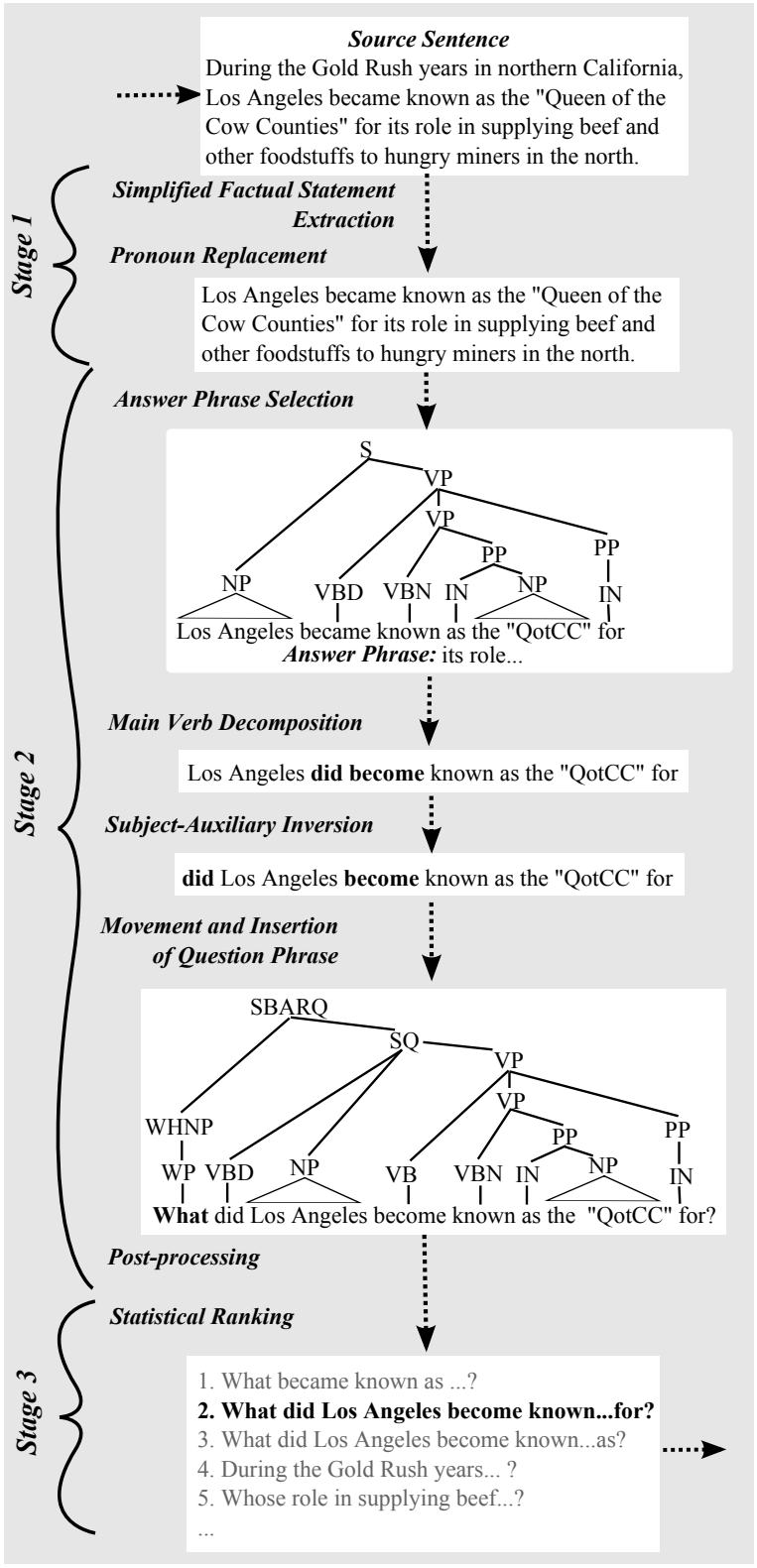

Figure 3.2: An illustration of the sequence of steps for generating questions, broken down into the three stages discussed in §3.2, §3.3, and §3.4. For clarity, parse trees are not shown for all steps. Also, while many questions may be generated from a single source sentence, only one path is shown.

our experiments with the end-to-end QG system. Such long sentences are relatively rare: only 2.1% of the sentences in the texts used for training in §4.5 of Chapter 4 exceeded 50 words.

3.1.2 The `Tregex` Tree Searching Language and Tool

We use the `Tregex` tree searching language (Levy and Andrew, 2006) to identify the relevant syntactic elements (e.g., subjects of sentences) as well as larger constructions (e.g., non-restrictive appositives) that our system operates on. `Tregex` allows us to match complex patterns in trees by combining various types of node-to-node relationships such as dominance and precedence and also nesting such relationships. Later, in §3.3.1, we provide an example `Tregex` expression for encoding a WH-movement constraint. See Levy and Andrew (2006) or the `Tregex` distribution for more information.

After identifying relevant parse tree nodes with `Tregex`, we manipulate trees using the Stanford Parser's API, which allows for inserting and deleting children, changing labels on tree nodes, etc.

3.1.3 Supersense Tagger

As part of mapping potential answer phrases to WH words (discussed in more detail in §3.3.2), we use a Java reimplementation of the supersense tagger described by Ciaramita and Altun (2006) to label word tokens with high-level semantic classes from WordNet (Miller et al., 1990), such as `noun.person`, `noun.location`, etc. The tagger labels proper nouns, as in the task of named entity recognition (Finkel et al., 2005), as well as common nouns and verbs. Supersense tagging can be viewed as a combination of named entity recognition, where just proper nouns are labeled with a small set of high-level semantic classes (often just `PERSON`, `ORGANIZATION`, and `LOCATION`), and word sense disambiguation (Yarowsky, 1995), where common nouns and verbs are labeled with very fine-grained word senses.

Since the supersense tagger is capable of telling us both that a proper noun such as *Thomas Jefferson* would lead to a *who* question, and that a common noun such as *beach* might lead to a *where* question, it is more suitable for QG than a typical named entity recognition approach that just labels proper nouns (we provide experimental evidence of this in §4.3 of Chapter 4).

The supersense tagger is a discriminative hidden Markov model (Collins, 2002) trained on la-

beled data from the SemCor corpus.[3] It uses features of both the current word (e.g., part of speech, capitalization features, the word itself) and the surrounding words to make predictions. Importantly, it also considers what the most frequent supersense in WordNet is for the current word as an additional feature. Ciaramita and Altun (2006) provide more details on the feature set. In the related work on word sense disambiguation, simply predicting the most frequent sense for each word is a very difficult approach to beat (McCarthy et al., 2004). However, in experiments with held-out data, Ciaramita and Altun (2006) found that the supersense tagger outperforms the most frequent sense approach 77% to 66% in terms of F_1 score. They also found that its performance for `noun.person`, `noun.location`, and `noun.group` was close to state-of-the-art named entity recognition systems that focus just on proper nouns in those semantic categories.

Below is an example of a sentence labeled with supersense tags.

B-noun.person	*I-noun.person*	*B-verb.social*	*B-noun.location*	*O*
Richard	Nixon	visited	China	to
B-verb.change	*B-noun.communication*	*O*		
improve	diplomacy	.		

As discussed in §3.3.2, the system uses these tags to create WH phrases (e.g., *who* for *Nixon*). The `B-` indicates the start of a span, and the `I-` indicates a continuation of a span. Our system does not use this distinction, instead using parse trees to identify phrasal boundaries. Also, our system only uses noun labels. The `O` label is for "other" tokens that are not nouns or verbs.

3.1.4 ARKref Noun Phrase Coreference Tool

We also use the ARKref coreference system[4] to identify the antecedents of pronouns. ARKref is a tool for noun phrase coreference that is based on the system described by Haghighi and Klein (2009).[5] It is a deterministic system that uses syntactic information from the Stanford Parser (Klein and Manning, 2003) and semantic information from an entity recognition component to constrain the set of possible mention candidates (i.e., noun phrase nodes) that could be antecedents for a given mention. It encodes syntactic constraints such as the fact that the noun phrases in predicative nom-

[3] The SemCor corpus is available at `http://www.cse.unt.edu/~rada/downloads.html`.

[4] The ARKref system was developed by Brendan O'Connor and Michael Heilman. See `http://www.ark.cs.cmu.edu/arkref/` for more details, and for a downloadable version of the system.

[5] At the time of writing, Haghighi and Klein (2009) had not publicly released their system.

inative constructions corefer (e.g., *John was the teacher.*), as well as semantic constraints such as the fact that *he* cannot corefer with a noun labeled as a location. After applying these constraints to the set of candidate antecedent phrases for a given mention, it selects as the antecedent the candidate node with the shortest tree distance from the target. To compute tree distance, the parse trees for all sentences are conjoined to form a single, right-branching document tree. Then, the syntactic tree distance between two nodes is defined as the number of nodes that exist on the path between them.

There are a few key differences between ARKref and the system from Haghighi and Klein (2009):

- ARKref does not include the bootstrapped lexical semantic compatibility subsystem referred to as `SEM-COMPAT` by Haghighi and Klein (2009).
- ARKref uses the supersense tagger described in §3.1.3 rather than a named entity recognizer to match entity types, allowing it to incorporate additional information common nouns.
- ARKref encodes a few additional constraints on coreference not addressed by Haghighi and Klein (2009).[6]
- ARKref uses male and female name lists from the 1990 U.S. Census[7] to identify when noun phrases refer to people by name (useful for resolving *he* and *she*). It also maps personal titles to genders when possible (e.g., to resolve *her* to *Mrs.*).

The ARKref tool demonstrates competitive performance in experiments with the ACE Phase 2 coreference dataset, referred to as `ACE2004-ROTH-DEV` by Haghighi and Klein (2009). It achieved 60.0% pairwise F_1 (65.8% precision and 55.2% recall), while Haghighi and Klein (2009) report 58.5% pairwise F_1 (68.2% precision and 51.2% recall. See Haghighi and Klein (2009) for more details on the dataset and evaluation metrics.

Example 3.12 provides an illustration of ARKref's output, in which brackets denote the extent of noun phrases and indices denote the entity to which each noun phrase refers.

[6]The additional constraints encoded in ARKref are the following: that objects cannot refer to subjects unless they are reflexive (e.g., in *John called him*, the pronoun *him* cannot refer to *John*); and that subjects cannot refer to sentence-initial adjuncts (e.g., in *To call John, he picked up the phone*, the subject *he* cannot refer to *John*). Binding theory (Chomsky, 1981; Carnie, 2006, Chapter 4) provides general explanations that explain these constraints as well as others not encoded in the system.

[7]The census name lists can be downloaded at `http://www.census.gov/genealogy/names/`.

(3.1) $[\text{John}]_1$ bought $[\text{himself}]_1$ $[\text{a book}]_2$. $[\text{Fred}]_3$ found out that $[\text{John}]_1$ had also bought $[\text{himself}]_1$ $[\text{a computer}]_4$. Next, $[\text{he}]_3$ found out that $[\text{John}]_1$ had not bought $[\text{him}]_3$ $[\text{anything}]_5$.

One can also interpret the output of a coreference tool as an undirected graph with nodes for each noun phrase and edges between noun phrases that refer to the same entity.

3.2 Stage 1: Transformations of Declarative Input Sentences

This section describes the first stage of our QG framework: the transformation of complex declarative input sentences into simpler factual statements that can be more readily converted into questions.

In our implementation, we include operations for two types of transformations: the extraction of sets of simplified factual statements from embedded constructions in complex input sentences, and the replacement of pronouns with their antecedents. We perform the simplified statement extraction first, and then replace pronouns in the extracted statements.

3.2.1 Extracting Simplified Factual Statements

Sentences often convey not just one but many pieces of factual information through the use of nested syntactic constructions.[8] As discussed in §2.2.2 of Chapter 2, we wish to take complex sentences such as *Prime Minister Vladimir V. Putin, the country's paramount leader, cut short a trip to Siberia* . . . , and extract simpler statements such as *Prime Minister Vladimir V. Putin cut short a trip to Siberia*. The task is essentially to generate a restricted set of entailed sentences, using the informal definition of entailment from the recognizing textual entailment task (Giampiccolo et al., 2007; Bar-Haim et al., 2007).

Our method for extracting meaning-preserving, simplified sentences depends on two linguistic phenomena: semantic entailment and presupposition. We provide some discussion of these phenomena to motivate our approach. For more detailed discussions, see MacCartney (2009) and Levinson (1983, Chapter 4).

[8] This section is taken from previously published work by Heilman and Smith (2010a).

Extraction and Simplification by Semantic Entailment

Semantic entailment is defined as follows: A **semantically entails** B if and only if for every situation in which A is true, B is also true (Levinson, 1983, Chapter 4). This section describes how we extract simplifications from complex sentences by removing adjunct modifiers and discourse connectives and by splitting conjunctions of clauses and verb phrases. These transformations preserve the truth conditions of the original input sentence, while producing more concise sentences from which questions can be generated.

Removing Discourse Markers and Adjunct Modifiers Many sentences can be simplified by removing certain adjunct modifiers from clauses, verb phrases, and noun phrases. For example, from example 3.2, we can extract example 3.3 by removing the discourse marker *however* and the relative clause *which restricted trade with Europe*.

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

(3.3) Jefferson did not believe the Embargo Act would hurt the American economy.

Example 3.3 is true in all situations where example 3.2 is true, and is therefore semantically entailed. Discourse markers such as *however* do not affect truth conditions in and of themselves, but rather serve to inform a reader about how the current sentence relates to the preceding discourse. Adjunct modifiers do convey meaning, but again do not affect semantic entailment. Of course, many adjuncts provide useful information that we should preserve for later QG steps. For example, prepositional phrases that identify the locations and times at which events occurred can be the target of *where* and *when* questions, respectively.

Our algorithm (which is presented later in this section) extracts simplified, entailed statements by removing the following adjuncts and discourse markers:

- non-restrictive appositives (e.g., *Jefferson, the third U.S. President, ...*)[9]

[9] Appositives and relative clauses can be restrictive or non-restrictive. Restrictive versions resolve referential ambiguities and are not typically offset by commas (e.g., *the man who was tall ...*). The non-restrictive ones provide additional information and are offset by commas (e.g., *John, who was tall, ...*). Our implementation only extracts non-restrictive forms, and distinguishes them from restrictive forms by including commas in the `Tregex` tree searching rules.

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

The system only removes verb phrase modifiers that are offset by commas For example, it simplifies example 3.4 to produce 3.5, but it does not simplify 3.6.

(3.4) When I talked to him, John was busy.

(3.5) John was busy.

(3.6) John was busy when I talked to him.

Also, the system does not remove the modifier *early* from the sentence *John did not arrive early* because that sentence does not imply *John did not arrive*. The effects on entailment of removing modifiers other than those offset by commas is a challenging question. It may be useful for QG to drop additional modifiers from very long sentences, but we leave that task to future work. Other work on textual entailment, such as the work by MacCartney (2009) may provide a better understanding of such phenomena.

Splitting Conjunctions We also split conjunctions of clauses and verb phrases, as in examples 2.24 and 2.25 from §2.2.2 of Chapter 2 about Vladimir Putin.[10] In most cases, the conjuncts in these conjunctions are entailed by the original sentence. For example, given *John studied on Monday but went to the park on Tuesday*, both *John studied on Monday* and *John went to the park on Tuesday* are entailed. Exceptions where conjuncts are not entailed include the following: conjunctions with *or* and *nor*, which we do not split; and conjunctions within downward monotone contexts such as negations or non-factive verbs, which we do split (we leave proper handling of this relatively rare case

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

to future work). An example of a sentence with a conjunction in a downward monotone context is example 3.7, which does not imply example 3.8.

(3.7) John did not run and swim.

(3.8) John did not run.

See MacCartney (2009, Chapter 6) for a discussion of downward monotone contexts.

Extraction by Presupposition

In addition to the strict notion of semantic entailment, the pragmatic phenomenon of presupposition plays an important role in conveying information. The semantically entailed information in complex sentences often covers only a fraction of what readers understand. Consider again the example about the Embargo Act:

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

If our algorithm were to only extract semantically entailed statements as discussed previously, then it would miss possibly useful questions about facts such as example 3.10 because they are not semantically entailed by example 3.9 (note how *the Embargo Act would hurt the American economy* is also not entailed).

(3.10) The Embargo Act restricted trade with Europe.

On the other hand, if an algorithm were to extract from all nested constructions and naïvely copy features such as negation from the main clause, then it would output many false sentences, such as example 3.11 (from the clause *which restricted trade with Europe*).

(3.11) The Embargo Act did not restrict trade with Europe.

As the examples show, some of the information in certain syntactic constructions is not semantically entailed but rather presupposed, or assumed to be true and stated indirectly, by the author

(Levinson, 1983, Chapter 4). The meaning conveyed by these constructions is not affected by non-factive verbs like *believe* and *doubt*, negation, modal operators, and other constructions that affect the meaning of the main clause of the sentence. This phenomenon of presupposition is often subsumed by the term "conventional implicature" (Levinson, 1983, Chapter 4).

Many presuppositions have clear syntactic or lexical associations, or "triggers." These triggers facilitate the extraction of simple statements that can lead to useful, concise questions. A list of presupposition triggers is provided by Levinson (1983, Section 4.2). The algorithm we present later in this section uses a subset of these presupposition triggers to extract simplified sentences from the following constructions (for each example, our method will extract the sentence *Jefferson was the third U.S. President*):

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

Note that this set of presupposition triggers is a subset of the adjunct modifiers that our method removes when simplifying sentences. It covers only subset of the adjunct modifiers because it was not clear how to create reliable and concise extraction rules for all of them. For example, extracting from certain parenthetical phrases, such as in *Jefferson (1743–1826)*, would likely require components to identify time expressions and to generate appropriately formatted output. We leave such extensions to future work. Also note that while we described how these modifiers are removed to simplify sentences, our algorithm performs extractions prior to removing modifiers.

We do not address all the possible triggers since many are rare, challenging, or unlikely to be useful for QG. For example, one trigger we do not include is that definite descriptions are taken to imply that the thing they refer to actually exists. The noun phrase *Thomas Jefferson* presupposes statements such as *There was a person named Thomas Jefferson*, but such statements are unlikely to lead to useful questions in (e.g., *Was there a person named Thomas Jefferson?*). Also, we leave to future work the challenging task of extracting from clauses that serve as complements to change

Algorithm 1 `extractSimplifiedSentences`(t)

$T_{result} \leftarrow \emptyset$
$T_{extracted} \leftarrow \{t\} \cup$ **extract** new sentence parse trees from t for the following: non-restrictive appositives; non-restrictive relative clauses; subordinate clauses with a subject and finite verb; and participial phrases that modify noun phrases, verb phrases, or clauses.
for all $t' \in T_{extracted}$ **do**
  $T_{result} \leftarrow T_{result} \cup$ `extractHelper`(t')
end for
return T_{result}

Algorithm 2 `extractHelper`(t)

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

of state verbs, factives, and implicatives (e.g., *John remembered that Jefferson was the third U.S. President*).

Extraction Algorithm

This section presents our algorithm for extracting simplified factual statements from complex input sentences. The algorithm operates on standard Penn Treebank-style (Marcus et al., 1993) phrase structure trees, as returned by the Stanford Parser (Klein and Manning, 2003). The primary method, `extractSimplifiedSentences`, shown in high-level pseudo-code in Algorithm 1, takes a tree

t as input and returns a set of trees T_{result} as output. A helper function, shown in Algorithm 2, recursively splits conjunctions and checks to make sure that outputs have subjects and finite main verbs.[11]

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

After parsing with the Stanford Parser (Klein and Manning, 2003) and identifying key nodes with `Tregex`, we manipulate trees using the Stanford Parser's API, which allows for inserting and deleting children, changing labels on tree nodes, etc.[12]

We include an additional filtering step in the extractor to avoid processing parse trees that appear to be ungrammatical sentences (e.g., image captions that are only sentence fragments), or sentences with unusual constructions. If the parse tree for an input sentence contains any of the following nonterminal symbols, the extractor returns a set containing just the unmodified tree as its output: `UCP` (unlike coordinated phrase), `FRAG` (sentence fragment), `X` (unknown constituent), `NAC` (not a constituent). Our implementation also returns just the unmodified tree if no simplified outputs are extracted by Algorithm 1 (e.g., due to the lack of a subject or finite main verb).

Relevant Work on Text Simplification and Compression

Many approaches to sentence compression and simplification have been proposed; see (Dorr and Zajic, 2003; Clarke, 2008) for reviews of various techniques. One particularly closely related method is discussed by Beigman Klebanov et al. (2004). That method extracts simple sentences from each verb in the syntactic dependency trees of complex sentences. However, the authors do not provide code or evaluate their system on realistic texts, so it is unclear how robust and effective their approach is (in a small evaluation with one text, 55% of the simplifications extracted by their system were acceptable).

[11] In our experiments, when checking whether a sentence had a subject and finite main verb, we also excluded statements whose main verb matched the regular expression `(includ.*|accord.*)`. This helps us avoid extracting rather uninteresting outputs such as *Fruits include apples* from *John likes fruits, including apples.*

[12] For example, the following rule matches appositive constructions: `NP < (NP=noun !$-- NP $+ (/,/ $++ NP|PP=appositive !$CC|CONJP)) >> (ROOT <<# /^VB.*/=mainverb)`. For each match, the system creates a new sentence of the form, "`noun` *be* `appositive`." The form of *be* depends on the part of speech of the main verb, `mainverb`, and the number and person of `noun`. See the paper on `Tregex` by Levy and Andrew (2006) for details about the rule syntax.

Recently, Zhu et al. (2010) describe a simplification technique involving a synchronous grammar over a set of simplification and paraphrasing operations. They train their model on a parallel corpus of complex and simple texts that was automatically extracted from the English and Simple English version of Wikipedia. However, it is not clear that such techniques provide fluent output—particularly since the automatically extracted training data are noisy.

For an example of extracting simplified statements from an extremely complex sentence, see Appendix E.

3.2.2 Pronoun Resolution

If extracted statements contained unresolved pronouns, they would almost always lead to vague questions since the question appears outside of its original discourse context (e.g., $\otimes$ *When was he elected President of the United States?*).

In order to avoid these vague questions, we replace pronouns with their first-occurring antecedent in the discourse (i.e., in the input text). The intuition behind this approach is that the the first antecedent is likely to be a specific and informative noun phrase since it was the first mention of the entity in the discourse. This simple method seems to work fairly well, although the first mention is not always the most informative (e.g., as in the sentence *An avid reader, Susan is always at the library*).

Pronoun replacement proceeds as follows. First, using the ARKref coreference system, we construct a coreference graph for all noun phrase mentions in the input text. Then, for each pronoun in the set of simplified extracted statements:

1. We identify the parse tree node in the original input sentence that corresponds to the given pronoun.
2. Then, using the coreference graph, we identify the subtree for the first mention that is an antecedent of the pronoun.
3. We then replace the pronoun in the extracted statement with a copy of its antecedent.

A few modifications account for special cases:

- If an antecedent mention has an appositive, non-restrictive relative clause, or a constituent in parentheses (e.g., as in *Harry Truman (1884-1972)*), then we select the child subtree that includes the head word.[13] This helps to create more concise canonical mentions (e.g., *Harry Truman* from *Harry Truman, who succeeded Franklin D. Roosevelt,*). For appositives like *the famous painter, Pablo Picasso*, where the identified syntactic head is not a proper noun, we choose the proper noun instead of the subtree containing the head.
- If a pronoun is possessive, then the replacement will also be made possessive if necessary by adding *'s*. If the pronoun is not possessive but the replacement is, then *'s* will be removed.
- When nesting occurs—that is, when the first mention of an entity contains mentions of other entities—we recursively replace mentions. For example, from the last sentence in *John likes cars. His car is expensive. It has leather seats.*, we extract *John's car has leather seats*.
- Within a sentence, only the first mention of an entity can be replaced, leading to *John thought he could win* rather than *John thought John could win*.
- If an indefinite determiner (*a* or *an*) modifies the antecedent, we replace it with the definite determiner *the*, which conveys that the noun it modifies is old information (we observe that questions such as *Who did the man meet?* are usually more natural than those such as *Who did a man meet?*).
- If the coreference tool could not identify an antecedent, we make no replacement and mark the current statement as having an unresolved pronoun. Stage 2 filters out questions containing unresolved pronouns unless they are part of the answer. For example, if stage 2 receives *he won the game*, it will generate *Who won the game?*, but not *What did he win?*, which we filter out (§3.3.6).

As a simple example of the pronoun replacement step, consider processing the last sentence of the following passage, which was previously discussed in §3.1.4.

(3.12) [John]$_1$ bought [himself]$_1$ [a book]$_2$. [Fred]$_3$ found out that [John]$_1$ had also bought [himself]$_1$ [a computer]$_4$. Next, [he]$_3$ found out that [John]$_1$ had not bought [him]$_3$ [anything]$_5$.

[13] Head words are identified using rules from Collins (1999).

Before pronoun replacement, the simplified factual statement extraction step will remove the adjunct *next* from the last sentence. Then, for the pronouns *he* and *him*, the ARKref tool identifies that *Fred* is the first antecedent (preferring *Fred* over *John* due to syntactic proximity). The pronoun *he* is replaced since it is the first mention of the entity Fred, but *him* is not replaced since it is the second mention. Finally, this step will output the statement *Fred found out that John had not bought him anything*.

Our approach to pronoun replacement is similar to the method described by Nenkova (2008) for replacing noun phrases with more informative ones in multi-document summarization. However, their method uses a simpler coreference resolution technique, and selects antecedents in order to improve a summarization quality heuristic.

One might also consider going beyond pronouns and replacing any noun phrase with its first antecedent. From some preliminary experiments, it appears that doing so would introduce more errors than improved questions because of the low performance of general noun phrase coreference. However, replacing just pronouns still allows us to generate questions about sentences that would otherwise be skipped or lead to vague questions.

The above process is quite complex and knowledge-intensive. It would be interesting to explore automatic techniques and the possibility of learning such pronoun resolution techniques from labeled data. We leave this to future work.

3.3 Stage 2: Question Creation

Stage 2 of the system takes as input a declarative sentence and produces as output a set of possible questions.[14] It identifies the answer phrases that may be targets for WH-movement and converts them into question phrases. In the current system, answer phrases can be the following:

- noun phrases (e.g., *Thomas Jefferson*)
- prepositional phrases (e.g., *in 1801*)
- subordinate clauses (e.g., *that Thomas Jefferson was the 3rd U.S. President*)

[14] The main clauses of declarative sentences are marked as "S" in the Penn Treebank. We leave inverted clauses ("SINV") for future work.

The system does not generate questions from all of the phrases of these types. There are certain constraints on the set of possible answer phrases, which we discuss in §3.3.1. Also, note that the system does not generate questions about verb phrases (e.g., *What happened to Thomas Jefferson in 1801?*).

From the possible answer phrases, the system generates questions with the following question words: *who*, *what*, *where*, *when*, *whose*, and *how many*. We discuss this step in §3.3.2. The system could be extended to detect and transform other types of phrases to produce other types of questions (e.g., *how*, *why*, and *what kind of*) since the transformation from answer to question is achieved through a composition of general-purpose rules. This would allow, for example, the addition of a relatively simple rule to generate *what kind of* questions by building off of the existing rules for subject-auxiliary inversion, verb decomposition, etc. The system also generates yes-no questions, as discussed below.

For each sentence, many questions may be produced: there are often multiple possible answer phrases in a particular sentence, and multiple question phrases for each answer phrase. For example, from *John met Sally*, we generate *Who met Sally?*, *Who did John meet?*, and *Did John meet Sally?*.

Stage 2 aims to overgenerate grammatical, though not concise or specific, questions. The transformation rules in stage 2, which encode a substantial amount of linguistic knowledge, are applied in the following order (note that some of these steps do not apply in some cases):

1. Mark phrases that cannot be answer phrases, due, for example, to WH-movement constraints (§3.3.1).
2. Select an answer phrase, and generate a set of question phrases for it (§3.3.2).
3. Decompose the main verb (§3.3.3).
4. Invert the subject and auxiliary verb (§3.3.4).
5. Remove the answer phrase and insert one of the question phrases at the beginning of the main clause (§3.3.5).
6. Post-process to ensure proper formatting (§3.3.6).

Figure 3.3 illustrates this process. We use the term "question phrase" to refer to the phrase at the start of a question that includes a WH word (e.g., *who*, *how many*).

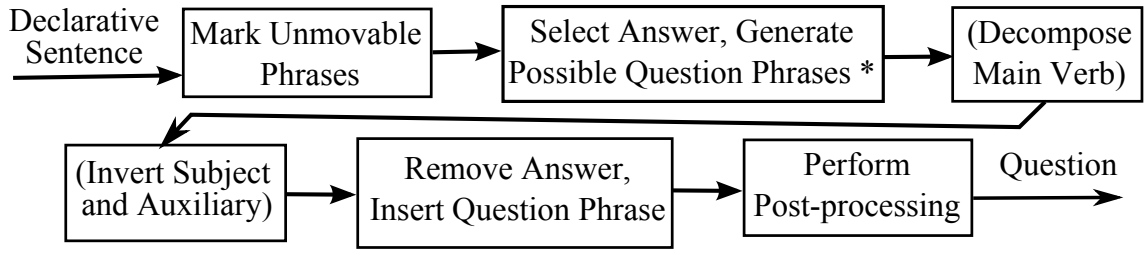

Figure 3.3: Process of transforming declarative sentences into questions in Stage 2. Parentheses mark steps which may not be necessary for certain questions. $*$ mark steps which may produce zero or multiple outputs.

There are two important special cases. First, when generating yes-no questions, there is no answer phrase to remove nor question phrase to insert. Second, when generating questions for which the answer phrase was the subject of the declarative sentence, decomposition of the main verb and subject-auxiliary inversion are not necessary. The subject is removed and replaced by a question phrase in the same position (e.g., *John met Sally* becomes *Who met Sally?*). Additionally, the main verb may need to be altered to agree with the question phrase if the subject is a first- or second-person pronoun (the method for finding different variants of verbs is described in §3.3.3.)

We now discuss each step in turn.

3.3.1 Marking Unmovable Phrases

A set of `Tregex` expressions is used to mark the phrases in an input tree which cannot be answer phrases due to constraints on WH-movement. Each parse tree node subject to a movement constraint is renamed by extracting the node's current label and adding a special prefix marker ("`UNMOVABLE-`") to its label. Then, in the subsequent step of answer phrase selection, these nodes are skipped so that the system does not generate questions for them.

For example, the following expression encodes the constraint that phrases under a clause with a WH complementizer cannot undergo WH-movement:

(3.13) `SBAR < /^WH.*P$/ << NP|ADJP|VP|ADVP|PP=unmv`

This expression matches particular types of "`SBAR`" subtrees. The operator "<" denotes that the

"SBAR" node should directly dominate (i.e., have as a child) a node whose label matches the regular expression "/^WH.*P\$/" (e.g., "WHNP" or "WHPP"). The operator "<<" and its argument denote that the "SBAR" should also dominate a node with a label that matches "NP," "ADJP," "VP," "ADVP," or "PP," and that this dominated node can be referred to by the name "unmv." When a match is found, any node with the name "unmv" can be identified and treated as unmovable due to this WH-movement constraint.

This constraint allows us to avoid generating the ungrammatical question **What did Darwin study how evolve?* from the sentence *Darwin studied how species evolve* (which was previously presented in example 2.34 from Chapter 2). Since the above expression matches, as illustrated in Figure 3.4, the system marks the noun phrase *species* as unmovable and avoids selecting it as an answer phrase.

The full list of 18 tree searching expressions for identifying potential answer phrases, which encode WH-movement constraints as well as a few other conservative restrictions in the system (e.g., not allowing answer phrases within direct quotations), is provided in Appendix A.

3.3.2 Generating Possible Question Phrases

After marking unmovable phrases, the system iterates over the possible answer phrases, generating possible questions for each one. This process also enables the extraction of answers to accompany each question, though we leave answer extraction to future work. Note that this step is skipped for yes-no questions since they have no question phrase.

To generate the question phrases, the system annotates the source sentence with a set of high-level semantic types using the supersense tagger, as described in §3.1.3. For a given answer phrase, the system uses these high-level semantic tags along with the syntactic structure to generate a set of one or more possible question phrases, each of which is used to generate a final question sentence.

Recall that answer phrases can be noun phrases (labeled "NP"), prepositional phrases (labeled "PP"), or subordinate clauses (labeled "SBAR"). For noun phrases, the system uses the conditions listed in Table 3.1 to decide what question phrases to generate. For subordinate clause answer phrases, the system only extracts the question phrase *what* (e.g., *What did John notice?* from *John noticed that the floor was wet*).

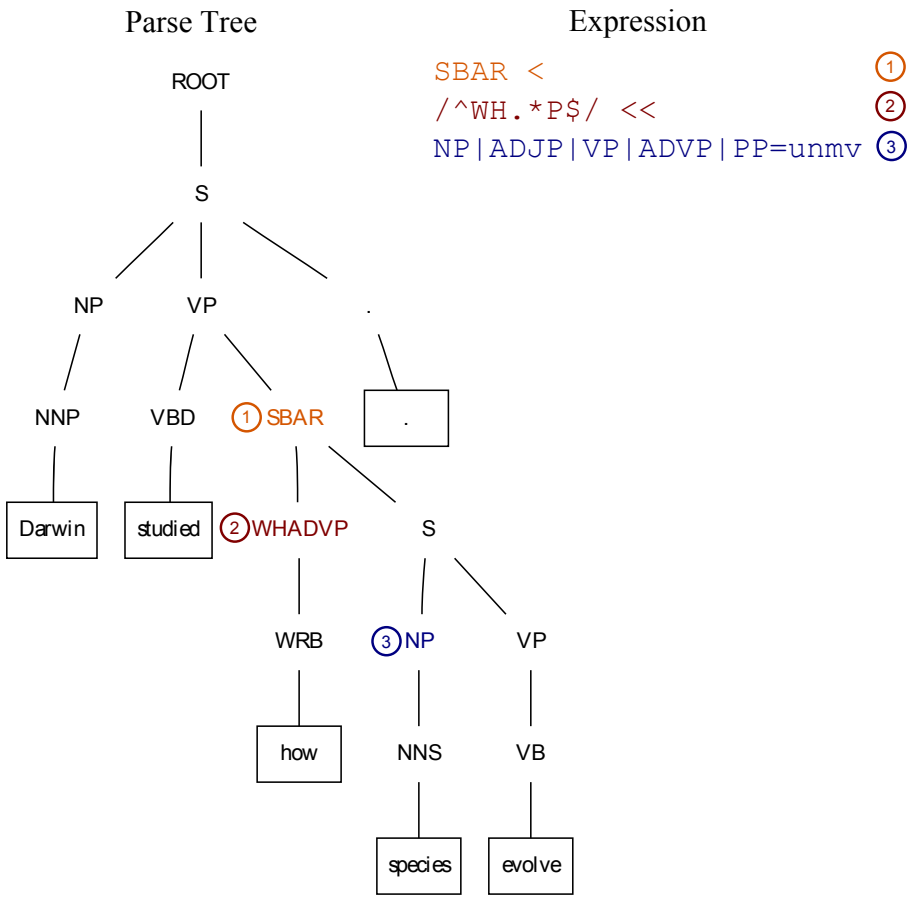

Figure 3.4: An example showing how a `Tregex` expression identifies a phrase (*species*) that is subject to a WH-movement constraint by matching a node in a sentence's parse tree. Indices indicate which nodes are matched by which parts of the expression.

For prepositional phrases, the system extracts the object of the preposition (a noun phrase) and uses the conditions listed in Table 3.1 to generate WH phrases from it.[15]

For noun phrases, there is one special case. For noun phrases with partitive constructions (e.g., *one of John's friends*), the question word selected by the system should correspond to the semantic type of the object of the adjoined prepositional phrase (e.g., *one of John's friends* should lead to a *who* question because *John's friends* refers to a set of persons). If the semantic type of the syntactic

[15] As discussed in §3.3.5, when the system inserts question words generated from prepositional phrases, it leaves prepositions in their original place (e.g., *Who did John give the book to?* from *John gave the book to Mary*), except for *where* and *when* questions (e.g., *Where did John find the book?* from *John found the book at the library*). Note that prescriptive grammar teachers might prefer moving the preposition (e.g., *To whom did John give the book?*), but doing so would likely lead to less natural-sounding questions in general.

WH word	Conditions	Examples
who	The answer phrase's head word is tagged `noun.person` or is a personal pronoun (*I*, *he*, *herself*, *them*, etc.)	*Abraham Lincoln*, *him*, *the 16th president*
what	The answer phrase's head word is not tagged `noun.time` or `noun.person`	*The White House*, *the building*
where	The answer phrase is a prepositional phrase whose object is tagged `noun.location` and whose preposition is one of the following: *on*, *in*, *at*, *over*, *to*	*in Japan*, *to a small town*
when	The answer phrase's head word is tagged `noun.time` or matches the following regular expression (to identify years after 1900, which are common but not tagged as `noun.time`): `[1|2]\d\d\d`	*Wednesday*, *next year*, *1929*
whose `NP`	The answer phrase's head word is tagged `noun.person`, and the answer phrase is modified by a noun phrase with a possessive (*'s* or *'*)	*John's car*, *the president's visit to Asia*, *the companies' profits*
how many `NP`	The answer phrase is modified by a cardinal number or quantifier phrase (`CD` or `QP`, respectively)	*10 books*, *two hundred years*

Table 3.1: The conditions in which various WH phrases are generated from a given answer phrase.

head were used, partitive constructions would incorrectly lead to *what* questions (e.g., *one* would lead to a *what* question). Thus, if the noun phrase is a partitive construction, the system extracts the object of the adjoined prepositional phrase in the partitive construction and generates WH phrases from it using the conditions in Table 3.1.[16]

It is worth noting that this step encodes a considerable amount of knowledge and yet still does not cover all relevant linguistic phenomena (see the error analysis in §4.11 of Chapter 4).[17] Methods for automatically learning a mapping from answer phrases to WH words seems an interesting area for future work. Prior work on question classification (Zhang and Lee, 2003; Metzler and Croft, 2005) may be relevant to learning such a mapping since question classification systems aim to predict the expected answer type for a given question—though, of course, question classification focuses on

analyzing questions rather than generating them.

3.3.3 Decomposition of the Main Verb

In order to perform subject-auxiliary inversion (§3.3.4), if an auxiliary verb or modal is not present, the system decomposes the main verb into the appropriate form of *do* and the base form of the main verb. It then modifies the tree structure of the verb phrase accordingly. For example, *John saw Mary* becomes *John did see Mary* before transformation into *Who did John see?* rather than **Saw John Mary?*.

If an auxiliary verb is already present, however, this decomposition is not necessary (e.g., *John has seen Mary* could lead to *Who has John seen?*). In such cases, the main verb phrase includes the auxiliary verb and a nested verb phrase containing the base form of the main verb.

The system identifies main verbs that need to be decomposed with the `Tregex` expression shown at the top of Table 3.2.[18]

In order to convert between lemmas of verbs and the different surface forms that correspond to different parts of speech, we created a map from pairs of verb lemma and part of speech to verb surface forms. We extracted all verbs and their parts of speech from the Penn Treebank (Marcus et al., 1993). We lemmatized each verb first by checking morphological variants in WordNet (Miller et al., 1990), and if a lemma was not found, then trimming the rightmost characters from the verb one at a time until a matching entry in WordNet was found. This simple approach works in practice for English because most verb forms either are derived from the lemma by adding a few letters (e.g., *-ed*) or are mapped to lemmas in WordNet's database.

[16] We identify partitive constructions using the following `Tregex` rule: `NP <<# DT|JJ|CD|RB|NN|JJS|JJR=synhead < (PP < (IN < of) < (NP <<# NN|NNS|NNP|NNPS=obj)) !> NP`. If the rule matches a given noun phrase, then the noun phrase is a partitive construction with a prepositional object `obj` if the terminal symbol under the node `synhead` is one of the following words: *series*, *set*, *number*, *part*, *more*, *all*, *none*, *rest*, *much*, *most*, *some*, *one*, *many*, *any*, *either*, *%*, *percent*, *portion*, *half*, *third*, *quarter*, *fraction*, *quarter*, *best*, *worst*, *member*, *bulk*, *majority*, *minority*. Otherwise, the phrase is not a partitive construction (e.g., *the department of state*). Note that this word list, which was produced by manually checking syntactic heads in partitive constructions that had been automatically extracted from the Penn Treebank (Marcus et al., 1993), is not exhaustive.

[17] One specific, relatively rare issue that the system does not address is that sentences about a person's role can lead to *what* questions in addition to *who* questions. For example, from the sentence *Mark is a farmer*, one would probably want to generate *What is Mark?* instead of or in addition to *Who is Mark?*.

[18] In the first expression in Table 3.2, for identifying the main verb, there is a disjunction of the form `[ A | B ]`. The first part (`A`) is for handling sentences with predicates that are not auxiliaries or are modified by auxiliaries (e.g., *John bought an apple*, *John has bought an apple*). The second part (`B`) handles sentences with predicates that can also be auxiliaries (e.g., *John has a book*).

Purpose	Expression
To identify the main verb, which the system decomposes into a form of *do* followed by the base form of the verb.	`ROOT < (S=clause < (VP=mainvp [ < (/VB.?/=tensed !< is|was|were|am|are|has| have|had|do|does|did) | < /VB.?/=tensed !< VP ]))`
To identify the main clause of for subject-auxiliary inversion.	`ROOT=root < (S=clause <+(/VP.*/) (VP < /(MD|VB.?)/=aux < (VP < /VB.?/=verb)))`
To identify the main clause for subject auxiliary inversion in sentences with a copula and no auxiliary (e.g., *The currency's value is falling*).	`ROOT=root < (S=clause <+(/VP.*/) (VP < (/VB.?/=copula < is|are|was|were|am) !< VP))`

Table 3.2: Tree searching expressions used for verb decomposition and subject-auxiliary inversion.

3.3.4 Subject-Auxiliary Inversion

The system performs subject-auxiliary inversion either when the answer phrase is a non-subject noun phrase or when the question to be generated is a yes-no question. For example, if generating questions from the sentence *Burr shot Hamilton*, subject-auxiliary inversion would be performed when generating questions 3.14 and 3.15, but not when generating question 3.16.

(3.14) Did Burr shoot Hamilton?

(3.15) Who shot Hamilton?

(3.16) Who did Burr shoot?

The system identifies the main clause and the relevant nodes in the verb phrase using the bottom two `Tregex` expressions in Table 3.2. The main clause node, initially "`S`", is first relabeled as "`SQ`", indicating that it is part of a question. Then the auxiliary or copula is moved so that it becomes the first child of this node. The "`SQ`" node is then used to form a new tree for the sentence. In the case of yes-no questions, the root node of the question tree will have the "`SQ`" node as its only child. In the case of WH-questions, the root node has an "`SBARQ`" node as its child. This "`SBARQ`" node then has the "`SQ`" node as a child, which will be preceded by a question phrase node (e.g., "`WHNP`").

After transforming the main clause and relabeling it "`SQ`," any leading adjunct phrases under this node and preceding a comma are moved to the front of the final sentence (e.g., to produce more

natural questions like *Following Thomas Jefferson, who was elected the 4th U.S. President?* rather than the more awkward *Who, following Thomas Jefferson, was elected the 4th U.S. President?*).

3.3.5 Removing Answers and Inserting Question Phrases

In the next step, the system removes the selected answer phrase, and for each possible question phrase generated from the answer phrase (as discussed in §3.3.2), it inserts that question phrase into a separate copy of the current tree to produce a new candidate question. The question phrase is inserted as a child of the "`SBARQ`" node under the root node, following any leading sentence-level adjunct phrases. Note that if the question is a yes-no question, then this step is not necessary.

If the answer phrase is a prepositional phrase, then only the object of the preposition is moved. This leads to questions such as *What is the building near?*, which we judged to be more natural than questions such as *Near what is the building?*. The only exception is *where* and *when* questions, for which the preposition is also removed (e.g., to generate *Where did John run?* rather than *In where did John run?* from the input sentence *John ran in the park*.).

3.3.6 Post-processing

Some additional post-processing mechanisms are necessary to ensure proper formatting and punctuation. Sentence-final periods are changed to question marks. Additionally, the output is de-tokenized to remove extra whitespace symbols (e.g., spaces preceding punctuation symbols).

We observed in preliminary output of our system that nearly all of the questions including pronouns were too vague (as discussed in §2.3.1 of Chapter 2). The system therefore filters out questions with noun phrases consisting solely of determiners (e.g., *those*) and noun phrases with unresolved pronouns. It also filters out questions that are over 30 tokens in length since such extremely long questions, though quite rare, are almost always awkward and unacceptable.

Note that the system does not skip questions with pronouns that were resolved to mentions within the same sentence and were thus not replaced during the pronoun replacement step in stage 1 (§3.2.2). For example, it does not filter out the question *What did John think he would win?* generated from the sentence *John thought he would win the prize* if the pronoun *he* refers to *John*.

3.4 Stage 3: Question Ranking

Stages 1 and 2 generate a large set of candidate questions, many of which are likely to be unacceptable. In particular, many questions may be vague due to the challenges discussed in §2.3.1 of Chapter 2.

Most users of QG tools will only consider a relatively short list of candidate questions, and so we want to be able to identify the questions that are most likely to be acceptable and present them first (i.e., at the top of a ranked list). Therefore, the stage 3 of the system ranks candidate questions from stage 2 using a statistical model of question acceptability.

In this section, we first discuss the statistical model and then describe the set of features it uses.

3.4.1 Statistical Model

We use least squares linear regression to model the quality of questions: we model quality as a linear function from a vector of feature values to a continuous variable ranging from 1 to 5, with higher numbers indicating higher levels of question acceptability according to linguistic factors such as grammaticality, vagueness, use of an appropriate WH word, etc.[19] For details on how we acquire labeled questions to train this model, see §4.4 in Chapter 4.

The acceptability y of a question x is modeled as follows:

$$y = \mathbf{w}^\top \mathbf{f}(x) \tag{3.17}$$

where $\mathbf{f}$ is a feature function that takes a question as input and returns a vector of real-valued numbers pertaining to different aspects of the question, and $\mathbf{w}$ is a vector of real-valued weight parameters for each feature in the feature vector returned by $\mathbf{f}$.

We seek a weight vector $\hat{\mathbf{w}}$ that minimizes the sum of the squared errors on the training data (we describe our datasets in §4.10 of Chapter 4), subject to a penalty on the ℓ_2 norm of the weight vector, to avoid overfitting:

[19] Heilman and Smith (2010b) used a logistic regression since they used a different question rating scheme that produced binary judgments judgments on a five-point scale. Note that both linear and logistic regression are instances of the generalized linear model (Nelder and Wedderburn, 1972).

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{N} \Big(y_i - \mathbf{w}^\top \mathbf{f}(x_i) \Big)^2 + \lambda ||\mathbf{w}||_2^2 \tag{3.18}$$

In the above equation, x_i is a single instance in the training data, with an acceptability value of y_i, and N is the total number of training examples. This form of penalized linear regression is often called ridge regression (Hastie et al., 2001). In our experiments, the regularization hyperparameter λ was selected through cross-validation on the training data.[20]

Later, in §4.7 of Chapter 4, we describe experiments that compare this ranking model to others.

3.4.2 Features

The features used by the ranker can be organized into several groups described in this section. This feature set was developed by an analysis of questions generated from development data. The numbers of distinct features for each type are denoted in parentheses, with the second number, after the addition symbol, indicating the number of histogram features (explained below) for that type. There are 179 features in total.[21]

Length Features (3 + 24)

The set includes integer features for the numbers of tokens in the question, the source sentence, and the answer phrase from which the WH phrase was generated. These numbers of tokens will also be used for computing the histogram features discussed below.

WH Words (7 + 0)

The set includes boolean features for whether the question was a WH question (rather than a yes-no question), and, if so, whether it contained each of the following WH phrases: *who*, *what*, *where*, *when*, *whose*, and *how many*.

[20] We use the WEKA toolkit (Hall et al., 2009) to train the linear regression model. WEKA standardizes each feature value by subtracting its mean and then dividing by its standard deviation (means and standard deviations are computed from the training data). After training is complete, WEKA converts the weights back to the original, unstandardized space.

[21] The feature set is slightly adapted from the feature set described by Heilman and Smith (2010b) to account for changes in stages 1 and 2.

Negation (1 + 0)

The set includes a boolean feature for the presence of *not*, *never*, or *no* in the question.

N-Gram Language Model Features (6 + 0)

The set includes real valued features for the length-normalized log likelihoods of the question, the source sentence, and the answer phrase. Separate likelihood features are included for unigram and trigram language models. These language models were estimated from the written portion of the American National Corpus Second Release (Ide and Suderman, 2004), which consists of approximately 20 million tokens, using Kneser-Ney smoothing (Kneser and Ney, 1995).

Grammatical Features (22 + 90)

The set includes integer features for the numbers of proper nouns, pronouns, adjectives, adverbs, conjunctions, numbers, noun phrases, prepositional phrases, and subordinate clauses in the phrase structure parse trees for the question and answer phrase. It also includes one integer feature for the number of modifying phrases at the start of the question (e.g., as in *At the end of the Civil War, who led the Union Army?*); three boolean features for whether the main verb is in past, present, or future tense; and one boolean feature for whether the main verb is a form of *be*.

Transformations (7 + 0)

The set includes boolean features for whether the following syntactic transformations were applied in stage 1 (§3.2.1) when creating the question: simplification by removal of modifiers, extraction from conjoined phrases, extraction from temporal subordinate clauses, extraction from participial phrases, extraction from non-restrictive relative clauses, and extraction from non-restrictive appositives.

It also includes a boolean feature for whether the answer phrase was the subject of the input sentence from stage 1, which encodes whether verb decomposition and subject-auxiliary inversion were performed (§3.3.3 and §3.3.4).

Vagueness (3 + 15)

The set includes integer features for the numbers of noun phrases in the question, source sentence, and answer phrase that are potentially vague. We define this set to include pronouns as well as common nouns that are not specified by a subordinate clause, prepositional phrase, or possessive. In the training data, we observed many vague questions resulting from such noun phrases (e.g., *What is the bridge named for?*). These features would be active for a noun phrase like *the president* but not more specific phrases such as *Abraham Lincoln* or *the U.S. President during the Civil War*.

Pronoun Replacement (1 + 0)

The feature set includes one boolean feature that is active when a pronoun was replaced with an antecedent during stage 1 (§3.2.2).

Histograms

In addition to the integer features for lengths, counts of grammatical types, and counts of vague noun phrases, the set includes binary "histogram" features for each length or count. These features indicate whether a count or length exceeds various thresholds: 0, 1, 2, 3, and 4 for counts; 0, 4, 8, 12, 16, 20, 24, and 28 for lengths. We aim to account for potentially non-linear relationships between question quality and these values (e.g., most good questions are neither very long nor very short).

3.4.3 Example

In this section, we provide a simplified example to illustrate how the ranking process works. Consider taking as input a text containing the following two input sentences.

(3.19) Lincoln was born in Kentucky.

(3.20) Lincoln faced many challenges during his presidency.

From these sentences, the first two stages of the QG system generate the following two questions, among others.

(3.21) Where was Lincoln born? (Kentucky)

(3.22) What did Lincoln face during his presidency? (many challenges)

Let us refer to question 3.21 as Q_{born} and question 3.21 as Q_{faced}. For simplicity, let us also assume there are only two features with the following weights:

- f_{proper}: the number of proper nouns in the answer (with weight $w_{proper} = 0.1$)
- f_{what}: a zero-one feature for whether the question contains *what* (with weight $w_{what} = -0.2$)

The model will also have a bias, or intercept, feature for a question's base score. This feature is always active. Let us assume its weight is $w_{bias} = 3.0$. For each question, the ranker extracts feature values, multiplies these by their corresponding weights, and sums up the results to produce a score.

$$\begin{aligned}\text{score}(Q_{born}) &= \mathbf{w}^\top \mathbf{f}(Q_{born}) \\ &= (w_{proper} * f_{proper}(Q_{born})) + (w_{what} * f_{what}(Q_{born})) + (w_{bias} * 1.0) \\ &= (0.1 * 1.0) + (-0.2 * 0.0) + (3.0 * 1.0) = 3.1\end{aligned}$$

$$\begin{aligned}\text{score}(Q_{faced}) &= \mathbf{w}^\top \mathbf{f}(Q_{faced}) \\ &= (w_{proper} * f_{proper}(Q_{faced})) + (w_{what} * f_{what}(Q_{faced})) + (w_{bias} * 1.0) \\ &= (0.1 * 0.0) + (-0.2 * 1.0) + (3.0 * 1.0) = 2.8\end{aligned}$$

The ranker then ranks the questions by their scores in descending order, such that the question Q_{born}, which asks about a specific location (*Kentucky*) will be ranked ahead of Q_{faced}, which has a rather vague answer phrase (*many challenges*).

In this way, the ranker enables the system to encode soft preferences for output questions that have certain types of features. Moreover, we can learn the weights for these features from labeled data so that we do not need to manually assign preferences ourselves (e.g., for whether the *who* questions from stages 1 and 2 are more likely to be acceptable than the *what* questions).

3.5 Summary

This chapter described our multi-stage framework for factual QG, and our implementation of that framework. The framework, designed to address some but not all of the challenges described in Chapter 2, consists of two rule-based stages that extract factual statements and convert them into candidate questions, followed by a statistical ranking step to allow the system to show better questions to users first.

The system encodes knowledge about the complex linguistic phenomena involved in question formation (e.g., subject-auxiliary inversion). It also applies machine learning techniques to make the system more robust and to model phenomena that are difficult to encode manually (e.g., whether a question is vague or awkward).

In order to facilitate future research on factual QG by other researchers, the code for our system is available for download at `http://www.ark.cs.cmu.edu/mheilman/questions`.[22]

Next, we present experiments that test the performance of the components of our QG system.

[22] Previously, we have shared code for our system with researchers from the Open University and the Institute for Creative Technologies at the University of Southern California, who have used the code in larger applications requiring automatically generating questions.

Chapter 4

Experiments

In the previous chapter, we described a framework and an implemented system for factual question generation. In this chapter, we discuss experiments that intrinsically evaluate the quality of the questions produced by the system.

We present experiments of two types. First, we present experiments that evaluate individual components of the system:

- §4.2 describes an evaluation of the simplified statement extraction component in stage 1 of the system (§3.2.1 of Chapter 3).
- §4.3 describes an experiment testing the ability of the supersense tagger (§3.1.3 of Chapter 3) to identify the high-level semantic types used in generating WH phrases.
- §4.7 describes experiments that test the question ranker (§3.4 in Chapter 3) and compare its performance to alternative ranking models. Then, §4.8 discusses some further analyses of ranking performance.

Second, after those component-wise experiments, we evaluate the end-to-end QG system by measuring the quality and quantity of top-ranked questions on held-out testing set articles, both at the document-level (in §4.10.1) and the sentence-level (in §4.10.2). These evaluations apply modified versions of the evaluation techniques for QG described by Heilman and Smith (2010b).

Also, in §4.11, we provide error analysis to identify the major sources of unacceptable questions from the system.

Later, in Chapter 5, we present a user study that extrinsically evaluates the system in the context of a tool for helping teachers create reading questions.

4.1 Text Corpora

In this section, we describe the corpora of texts used in the experiments. In later sections (§4.2, §4.5), we describe specific issues such as the number of articles from each source that were used in a particular experiment. Both corpora consist of primarily informational texts.

One set of texts is from the English Wikipedia,[1] which provides potentially noisy texts (e.g., with typographical and grammatical errors) about a wide variety of topics. We downloaded the set of "featured" articles in order to focus on topics of general interest. We also selected only the featured articles that were between 250 and 2,000 words in length (i.e., approximately 1-8 pages). When sampling Wikipedia articles for particular experiments, we manually filtered out some of the articles that were on obscure topics, such as 1980s video games.[2] We used the `wikiprep.pl` program[3] to extract plain text from Mediawiki's internal format.

The second set of texts is from the Encyclopedia Britannica and the Encyclopedia Britannica Elementary Edition.[4] It is comprised of articles about capital cities around the world. For each city, it contains two articles, one at an adult reading level and another at an elementary school level. When sampling texts for our experiments, we avoided using both of the articles about the same city. Also, this set of texts provides us with a well-edited source of texts from domain areas—namely, history and geography—that could be particularly fruitful for QG applications due to the prevalence of important dates, places, and people. It is also perhaps somewhat closer in style than Wikipedia articles to the sort of informational texts used in classrooms.

4.2 Simplified Statement Extraction Experiments

In this section, we investigate the following research questions:

[1] The English Wikipedia dataset was downloaded December 16, 2008.

[2] Wikipedia contains a surprising number of articles on Nintendo games from the 1980s.

[3] See `http://www.cs.technion.ac.il/~gabr/resources/code/wikiprep/` for details.

[4] The Britannica dataset has also been used in research on readability (Barzilay and Elhadad, 2003) and discourse processing (Barzilay and Lapata, 2008).

- Does the simplified factual statement extractor (§3.2.1 of Chapter 3) extract fluent and semantically correct factual statements from embedded constructions in complex sentences?
- How does the performance of the extractor compare to the performance of a similar sentence compression approach?

In this section, we evaluate the simplified statements produced by the extractor, not the questions they lead to (we evaluate performance of the end-to-end system in §4.10). To evaluate the extractor, we randomly sampled 25 articles from the set of Encyclopedia Britannica articles described in §4.1. For these experiments, we chose to focus on just the Britannica articles because they are informational texts written with fairly complex sentence structure—unlike the Britannica Elementary Edition articles—illustrating the benefits of the extraction techniques we describe. The Britannica articles are also somewhat better edited than Wikipedia articles, which may help us understand the performance of the extractor without the additional preprocessing errors that may arise from noisy texts.

The experiments in this section were previously presented by Heilman and Smith (2010a).[5]

4.2.1 Baselines

We compare to two baselines. The first baseline is the Hedge Trimmer (hereafter, HT) algorithm (Dorr and Zajic, 2003), a rule-based sentence compression algorithm that iteratively performs various simplification operations until an input is shorter than a user-specified target length. We decided to use HT rather than a statistical sentence compression system because Dorr and Zajic (2003) found that HT produces more fluent output (that paper compared HT to a hidden Markov model for compression). We implemented HT using `Tregex` rules, and made two modifications to adapt HT for QG: We set its target output sentence length to 15, rather than 10. In preliminary experiments with the target length set to 10, HT often led to ungrammatical output and seemed to drop information that would be useful for a QG application. Also, our implementation did not include the rules described by Dorr and Zajic (2003) for dropping determiners and temporal clauses (called "low-content units") because those rules are tailored to the headline generation application that Dorr and Zajic (2003) focused on.

[5] A few of the results from Heilman and Smith (2010a) were modified very slightly to correct for minor errors in calculation. For example, the average fluency for the main clause only baseline changed from 4.72 to 4.73. None of these changes affected the findings of Heilman and Smith (2010a).

The second baseline (MC-only) is to use our statement extractor but only to extract from the main clause (taking left most conjunct when conjunctions are present). This is an intermediate case between HT and our full extraction system: like HT, it only produces only one output per input. Also, note that the outputs will essentially be a subset of those from the full extractor.[6]

4.2.2 Experimental Setup

We ran our extractor and HT on the sample of 25 encyclopedia articles. All systems used the Stanford parser (Klein and Manning, 2003). Because parsing performance is typically quite poor for extremely long sentences, we set the maximum sentence length for the parser to be 60 word or punctuation tokens, and for the 3.1% of sentences that exceeded this maximum, all systems simply returned the input sentence as their output.

We also conducted a small-scale manual evaluation of the quality of the output. We randomly sampled 150 sentences from the encyclopedia articles. For 50 of these, we extracted using HT; for the remaining 100, we extracted using our method (which essentially subsumes MC-only).[7] When the full extraction system generated multiple outputs from an input sentence, we randomly sampled one of them.

We then asked two people not involved with this research to evaluate the 150 outputs for fluency and correctness, on 1 to 5 scales, following evaluation setups for paraphrase generation (Callison-Burch, 2007) and sentence compression (Knight and Marcu, 2000). In the instructions given to raters, a correct output sentence was defined as a sentence where at least part of the meaning of the input sentence was preserved (i.e., whether the output was true, given the input). Raters were shown each output sentence alongside its corresponding input sentence. The input-output pairs were randomly shuffled together, and the raters were blind to which system produced them. The Pearson correlation coefficients between the raters' judgments were $r = 0.92$ for fluency and $r = 0.82$ for correctness. For our analyses, we used the averages of the two raters' judgments.

[6]The output of the MC-only baseline is not completely subsumed by the output of the full system. For 1% of input sentences, the MC-only baseline outputs an unchanged copy of the input while the full system does not. This occurs when there are nested constructions for the full system to extract from, but no main clause with a subject and suitable finite main verb (as discussed in §3.2.1 in Chapter 3) for the MC-only system to extract from.

[7]For comparing our approach to the Hedge Trimmer system, we used 2 disjoint sets of sentences so individual raters would not compare extractions for the same sentence. We included 100 for the full extractor rather than 50 in order to get accurate estimates for MC-only baseline.

	Input	HT	MC-only	Full
Number of sentences extracted	2,952	2,952	2,952	4,820
Sentences per input	1.00	1.00	1.00	1.63
Mean Sentence length	23.5	12.4	15.7	13.4
Input word coverage (%)	100.0	61.4	71.7	87.9
Inputs unchanged (%)	100.0	33.8	27.9	29.7
Fluency	–	3.53	4.73	4.75
Correctness	–	3.75	4.71	4.67
Rated 5 for both Fluency and Correctness (%)	–	44.0	79.7	75.0

Table 4.1: Various statistics comparing the full extraction system to the baselines and the unmodified input sentences. All statistics are averaged across input sentences.

For a rough estimate of how much of the information in the input sentences was included in the outputs, we computed, for each system, the percentages of input word types, lemmatized with WordNet (Miller et al., 1990), that were included in at least one output. That is, we computed the following quantity, where N is the number of sample input sentences, s_i is a particular input sentence, $\mathbf{t}(s_i)$ is the set of lemmatized word types in s_i, and $\mathbf{e}(s_i)$ is the set of lemmatized word types in the outputs extracted from s_i:

$$\frac{1}{N}\sum_{s_i}^{N}\frac{|\mathbf{t}(s_i)\cap\mathbf{e}(s_i)|}{|\mathbf{t}(s_i)|}$$

4.2.3 Results

The results of our experiments show that the full extraction system generates more outputs per input than the two baselines on average (1.63 compared to 1.0), as shown in Table 4.1. While the main clause baseline constitutes a majority of the output of the full system (60.1%, including the sentences over 60 words that were not parsed), the other extraction rules contribute a substantial number of additional outputs, as shown in Table 4.2. For example, 27.6% of outputs involved splitting conjunctions, and 11.6% were extracted from appositives. Also, the systems do not unnecessarily alter sentences that are already short: 17.1% of the inputs for the full extractor were unchanged (e.g., *Brussels lies in the central plateaus of Belgium.*).

As shown in Table 4.1, the outputs from all systems were substantially shorter than the input sentences, which were 23.5 words long on average (not including punctuation). HT outputs were 12.4 words long on average, while outputs from the full extraction system were 13.4 words long on aver-

Operation	% of Outputs
Extraction from Conjunctions	27.6
Extraction from Appositives	11.6
Extraction from Non-Restrictive Relative Clauses	5.1
Extraction from Participial Modifiers of Noun phrases	4.3
Extraction from Temporal Subordinate Clauses	3.9
Extraction from Participial Modifiers of Verb phrases	1.6
Movement of Leading Prep. Phrases or Quotes	13.9
Removal of Appositives	13.8
Removal of Parentheticals	12.6
Removal of Non-Restr. Rel. Clauses or Participial Phrases	9.9
Removal of Verbal Modifiers after Commas	9.5
Removal of Leading Modifiers	7.0
Unparsed Sentences (over 60 words)	3.1

Table 4.2: Percentages of the extracted simple sentences that involved each simplification or extraction operation. The percentages exceed 100% because some outputs involved multiple operations.

age.[8]

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

The full extraction system appears to provide better coverage of the information in the input. 87.9% of input words appeared in at least one output from the full extraction system, compared to 61.4% for HT and 71.7% for the main clause baseline. The difference in coverage between the full system and the MC-only baseline, which provided the next best coverage, is statistically significant (Wilcoxon signed-rank test, $p < 0.001$).

In an informal analysis of the errors made by the systems, we observed that the HT baseline is often overly aggressive when compressing sentences: it frequently drops key function words and informative modifiers. For example, from Ex 2.20, it produces the ungrammatical sentence *Prime Minister Vladimir V. Putin cut short a trip returning oversee the federal response*, and the grammatical but terse sentence *Mr. Putin built his reputation*.

Most of the mistakes from the full extractor (and the MC-only baseline) appear to be the result of

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

parsing errors. For example, for the following input sentence, the parser incorrectly attaches *in 1851* to the main verb *started* rather than the modifier *opened to traffic*.

(4.1) Five years later work started on the railway to Moscow, opened to traffic in 1851.

The full extractor then extracts the following simplified version of the main clause, which was rated 5 fluency but 1 for correctness because it refers to an incorrect date (*in 1851*):

(4.2) Work started on the railway to Moscow in 1851.

4.2.4 Discussion

The experiment described in this section shows that the factual statement extractor extracts concise, fluent, and semantically correct statements from complex input sentences. The extractor also outperforms a simpler sentence compression baseline for this task.

It is worth pointing out that our approach aims for high precision: it does not aim to extract everything, and it does not paraphrase. A statistical model for this task might improve diversity and coverage, but it is unclear whether the output will be of similarly high fluency and correctness. For example, Zhu et al. (2010) developed a simplification model and trained in on a dataset of aligned corpora from English and Simple English Wikipedia. However, they did not acquire human judgments to measure fluency and correctness, instead relying on coarse automatic metrics such as BLEU (Papineni et al., 2002) to evaluate the quality of system outputs.[9]

As a general point, the problem of extracting simplified factual statements can be described as a form of textual entailment generation. Textual entailment has received considerable attention recently for the task of *recognizing* entailment (Dagan et al., 2005; Bar-Haim et al., 2007), but generating entailed sentences is also an important area for research. In fact, many text-to-text generation problems can also be viewed as entailment generation. Examples include paraphrase generation (Callison-Burch, 2007), sentence simplification (Zhu et al., 2010), sentence compression (Knight and Marcu, 2000; Clarke, 2008), and even, to some extent, QG. In all of these example tasks, one aims to generate output sentences that are true given some input text. The tasks vary in the additional restrictions that they place on those outputs (e.g., in sentence compression, the less important words

[9] Zhu et al. (2010) discuss various shortcomings of using machine translation metrics for their experiments and state that it is "not clear how well BLEU and NIST discriminate simplification systems."

in the input should be removed). Viewing text-to-text generation problems in this light may allow researchers to draw connections to existing research on semantics, pragmatics, and other linguistic phenomena, as we have done here (§3.2.1 of Chapter 3) Also, it may be possible to develop general approaches that work for multiple entailment generation problems: previous work has shown that general approaches can work for multiple entailment recognition problems (Heilman and Smith, 2010d; Chang et al., 2010).

4.3 Semantic Type Labeling Experiments

In this section, we investigate the performance of the supersense tagging component of our QG system, which is used in the process of mapping answer phrases to question phrases (§3.3.2 of Chapter 3). We ask the following research question:

- Does supersense tagging provide us with better high-level semantic information about answer phrases than two alternative methods?

More specifically, we are interested in whether supersense tagging can more accurately label the head words of noun phrases with tags that map well to WH words (e.g., answers tagged `noun.person` could lead to *who* questions).

4.3.1 Alternative Approaches to Semantic Type Labeling

We compare supersense tagging to two alternatives: a named entity recognizer (NER) and a tagger that labels words with their most-frequent supersense in WordNet (Miller et al., 1990). For supersense tagging, we use the re-implementation of the supersense tagger (SST) described by Ciaramita and Altun (2006) that is used in our QG system (§3.1.3). For an NER tool, we use the `2008-05-07` version of the Stanford NER system (Finkel et al., 2005), which tags words as either `PERSON`, `LOCATION`, `ORGANIZATION`, or `OTHER`.[10] For the most-frequent supersense alternative (MFS), we use the method described by Ciaramita and Altun (2006, §5.3). We deterministically map the tags from the

[10] We used the `ner-eng-ie.crf-3-all2006.ser.gz` model included in the Stanford NER distribution, which is available at `http://nlp.stanford.edu/software/stanford-ner-2008-05-07.tgz`. It is trained on data from the CoNLL, MUC6, MUC7, and ACE shared tasks. The distribution does not include a model that outputs date or time tags.

Common Tag	**NER Tag**	**MFS Tag**	**SST Tag**	**WH Word**
person	PERSON	noun.person	noun.person	*who*
organization	ORGANIZATION	noun.group	noun.group	*what*
location	LOCATION	noun.location	noun.location	*where*
date or time	$\emptyset$	noun.time	noun.time	*when*
other	OTHER	all except the above	all except the above	*what*

Table 4.3: The mapping from the tags produced by the NER, SST, MFS systems to the common tag set, along with the WH word most closely associated with each tag. See §3.3.2 of Chapter 3 for a full description of the method used in the QG system for mapping answer phrases to question phrases.

NER, SST, and MFS systems to a common tag set, roughly corresponding to WH words as shown in Table 4.3.

As discussed in §3.1.3 of Chapter 3, the NER tool uses a statistical model to incorporate contextual information but only tags proper nouns, while the MFS approach labels all types of nouns[11] but does not use contextual information. The SST tool, however, both tags all nouns and uses contextual information.

4.3.2 Evaluation

We randomly sampled 400 sentences from 20 Wikipedia articles from the corpus discussed in §4.1 (these same Wikipedia articles are used later in §4.10). Here, we chose to use just Wikipedia articles because we felt that semantic tagging experiments on Wikipedia would be more interesting to researchers working on NLP and information extraction, where Wikipedia is a common source of data (e.g., Hoffmann et al., 2010). Also, the Wikipedia articles likely involve a wider range of semantic types than the Britannica articles, which focus on cities.

We parsed each sentence with the Stanford Parser (Klein and Manning, 2003), and then randomly selected one of the noun phrases in the parse tree.[12] The author of this work then manually annotated the head word of each of the sampled noun phrases with one of the common tags in Table 4.3. These head word labels serve as a gold standard. To check inter-rater agreement, a second,

[11] The MFS and SST taggers also tag verbs, but we do not use those tags in our QG system.

[12] We selected noun phrases from the set of "maximal noun phrases," which are NP nodes in the parse tree that include all modifiers (e.g., relative clauses, prepositional phrases, determiners) of their head word, as done by Haghighi and Klein (2009), for example. We also automatically filtered out head words that were pronouns, possessives (i.e., *'s* and *'*), and the existential *there*.

Semantic Type	Count	NER	MFS	SST
`person`	70	0.34	0.46	**0.86**
`organization`	32	0.28	0.50	**0.53**
`location`	64	0.45	0.64	**0.75**
`date or time`	35	0.00	0.46	**0.57**
`other`	199	**0.97**	0.86	0.77
Overall	-	0.64	0.69	**0.75**
Macro-average	-	0.41	0.58	**0.70**

Table 4.4: Performance of different taggers at the task of predicting the high-level semantic types of noun phrase head words in Wikipedia texts, in terms of per-type, overall, and macro-averaged proportion accurate (i.e., correct). The "Count" columns shows the number of gold-standard tags for each type (out of 400 total). Results for the best performing system for each type are in bold.

independent annotator labeled a randomly selected subset of 200 of the head words. Agreement was high: 88% overall ($\kappa = 0.82$).

To evaluate the taggers, we compare the gold standard semantic category labels for the head words to the labels predicted by each system.[13] We measure performance in terms of overall (i.e., micro-averaged) accuracy and the accuracy for each semantic type (e.g., the number of correct `person` predictions divided by the number of gold standard `person` labels). We also compute the macro-average across the 5 categories, as in the following equation, where $\mathcal{Y}$ is the set of possible output labels, y is a particular label type, y_i is the true label for the ith instance, $\hat{y}_i$ is the predicted label for the ith instance, N is the number of test instances, and δ is a function that returns 1 if its argument is true and 0 otherwise:

$$\text{Macro-averaged Accuracy} = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} P(\hat{y}_i = y_i | y_i = y) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \frac{\sum_i^N \delta(y_i = y)\delta(\hat{y}_i = y)}{\sum_i^N \delta(y_i = y)}$$

This evaluation metric, which could also be called macro-averaged recall, accounts for the fact that some categories are more common than others, which a simple micro-average of accuracy does not. It is also somewhat simpler and perhaps more interpretable than macro-averaged F_1.

Proper Nouns

Semantic Type	Count	NER	MFS	SST
`person`	38	0.63	0.26	**0.95**
`organization`	20	0.45	0.40	**0.45**
`location`	45	0.64	0.67	**0.80**
`date or time`	6	0.00	**1.00**	**1.00**
`other`	22	0.73	**0.86**	0.32
Overall	-	0.60	0.56	**0.72**
Macro-average	-	0.49	0.64	**0.70**

Common Nouns

Semantic Type	Count	NER	MFS	SST
`person`	32	0.00	0.69	**0.75**
`organization`	12	0.00	**0.67**	**0.67**
`location`	19	0.00	0.58	**0.63**
`date or time`	29	0.00	0.34	**0.48**
`Overall`	-	0.66	0.75	**0.76**
Macro-average	-	0.20	0.63	**0.67**

Table 4.5: Performance of different taggers at the task of predicting the high-level semantic types, in terms of per-type, overall, and and macro-averaged proportion accurate (i.e., correct), broken down by whether the noun was a proper noun or not. The "Count" columns shows the number of gold-standard tags for each type. Results for the best performing system for each type are in bold.

4.3.3 Results

As shown in Table 4.4, the SST and MFS taggers substantially outperform the NER tagger, which is not surprising since the NER tagger does not label times or common nouns. Also, SST outperforms MFS, indicating that it is able to effectively use contextual information when making its predictions. The difference between the overall accuracy for SST and MFS is statistically significant (sign test, $p = 0.005$), as is the difference between overall accuracy for SST and NER (sign test, $p < 0.001$).

To explore why the SST tagger performs better, we can break down the results by whether noun phrases were headed by proper nouns or not, as shown in Table 4.5 (note that some of the resulting sample sizes are quite small, and so we avoid making strong conclusions here). Somewhat surprisingly, the SST and MFS systems perform quite well for proper nouns—in fact, the SST system produced better results for the `person` and `location` types than the NER system. We attribute

[13]This is somewhat different than typical NER evaluations, which compare the sets of multi-word sequences (i.e., not just head words) in the gold-standard to those extracted by an automatic tagger. In contrast, in this evaluation and in our QG application, we make use of syntactic parse information to extract syntactic heads and phrasal boundaries. Also, due to WH-movement constraints (3.3.1 of Chapter 3), not all the noun phrases sampled in this tagging evaluation would be answer phrases in our QG system. We felt that the relatively simple evaluation approach we chose (i.e., using a sample from all noun phrases) would make the results more interpretable and more generally useful.

the high performance on proper nouns to the SST and MFS taggers' use of WordNet (Miller et al., 1990), which, though focused on common nouns, contains quite a number of names. For example, for the text . . . *Cunningham enjoyed several ceremonial positions* . . . , the NER tagger predicts `other` for the name *Cunningham*, while the SST and MFS taggers correctly predict `person` because *Cunningham* is categorized as a person in WordNet.[14]

Also, while the performance of the SST and MFS systems is similar for proper and common nouns, the NER tagger performs quite poorly for common nouns because it always predicts `other`.

4.3.4 Discussion

While the SST system performs well, its average performance is still only around 70%, meaning that it will introduce errors into downstream applications such as our QG system. Here are a few examples that illustrate the sorts of errors made by the SST tagger.

For the following sentence, the tagger predicts the label `noun.artifact` for the word *rulers* (as in *measured with a ruler*) rather than `noun.person`:

(4.3) In December 1893, the Sikkimese monarchy and Tibetan rulers signed an agreement to increase trade between the two nations.

Next, for the following sentence, the SST tagger labels *age* as a `noun.attribute` (as in *the young child's age*), rather than a `noun.time`:

(4.4) The limestone of the Solnhofen area has also yielded such well-preserved fossils as Archaeopteryx with feather impressions and some pterosaurs with imprints of their wing membranes that are dated to the lower Tithonian age.

Finally, for the following sentence, the tagger incorrectly predicts `other` for the word *Nacala-a-Velha* rather than `noun.location`:

(4.5) The most severe damage was to generally poorly-built houses in Memba, Nacala-a-Velha, Mogincual, and Nampula city.

[14]The Stanford NER model does not make use of gazetteers or name lists, which some other NER taggers do (Carlson et al., 2009). For the SST and MFS taggers, WordNet can be viewed as a type of gazetteer.

Also, recall that tagging nouns with semantic types addresses only some of the challenges of mapping answer phrases to WH words (§2.1.1 of Chapter 2). For example, even if a noun phrase is correctly placed into the `other` category, asking a question with just *what* as its question phrase (e.g., rather than more specific phrases such as *what country* or *what month*) may lead to vague questions in certain discourse contexts. Later, in §4.11, we describe some errors that result even with correctly predicted semantic types.

In concluding this section, we observe that in addition to being relevant to our QG application, our evaluation could be useful in other NLP research (e.g., question answering research), for the following reasons:

- The evaluation metrics and methodology are not closely tied to our QG application. E.g., the categories `person`, `organization`, etc. are widely used.
- It directly compares NER to SST, which previous work has not done (Ciaramita and Altun, 2006), though our evaluation is biased against NER since it includes times and common nouns.
- It focuses on Wikipedia articles, which are out-of-domain and somewhat noisy compared to the news texts typically used to develop such taggers.

4.4 Question Rating Scheme

In subsequent sections, we describe experiments that involve all the components of our QG system. Before presenting results, we explain how we measure the acceptability of the ranked questions produced by the system.

First, we describe the rating scheme used to measure acceptability. The ratings focus on general linguistic factors such as grammaticality and vagueness, avoiding more complex issues such as whether a question is "interesting" or would make a good assessment for a particular type of learner (we leave such decisions to users, as discussed in Chapter 5).

Using the rating scheme, a set of human judges create labels for the acceptability of questions produced by the QG system, as discussed below. We use these labeled questions in two ways:

- to train the statistical model of question acceptability that the system uses to rank candidate questions (§3.4 of Chapter 3)

- to evaluate the final output of the system by measuring the quality of the top-ranked questions generated from held-out test articles.

Rating questions is a difficult task, and previous research has not developed annotation schemes that result in high levels of inter-rater agreement (Heilman and Smith, 2010b, 2009; Rus et al., 2010).

For example, Heilman and Smith (2009) used a fairly complex rating scheme involving 8 binary decisions for different possible deficiencies. Using Cohen's κ to measure agreement, they found agreement of $\kappa = 0.42$ for the overall boolean decision about whether any or none of the deficiencies were present. This level of agreement can be described as as "moderate," according to Landis and Koch (1977). Agreement was considerably lower for individual categories (e.g., $\kappa = 0.29$ for grammaticality).

4.4.1 Rating Scheme and Process

The rating scheme we use—which involves just a single, holistic judgment of question quality—was developed to be easy for novice annotators. It makes use of the following five-point holistic scale for question acceptability:

- **Good (5)** - The question is as good as one that a teacher might write.[15]
- **Acceptable (4)** - The question does not have any problems.
- **Borderline (3)** - The question might have a problem, but I'm not sure.
- **Unacceptable (2)** - The question definitely has a minor problem.
- **Bad (1)** - The question has major problems.

In judging whether questions exhibit problems, raters were instructed to focus on linguistic factors (grammaticality, the correctness of information, vagueness, awkwardness), following Heilman and Smith (2009). They were also asked to consider each question independently, such that similar questions about the same information would not affect each other's ratings. The full rating instructions are included in Appendix B.

[15]The phrase *as good as one that a teacher might write* is, of course, not particularly precise, but we felt it would make intuitive sense to novice annotators.

We created a simple web-based annotation interface to allow raters to view questions together with the articles from which they came. When rating a particular question, the interface would highlight the source sentence used in generating the question, for convenience. Raters were allowed to rate questions in any order and could alter previous ratings as they saw fit.

Before rating the questions used in our experiments, the raters worked through a practice article with twenty questions generated by the system. After these questions were rated, the author of this work provided feedback and guidance on the rating scheme as necessary.

In our experiments, in order to address imperfect inter-rater agreement, each question was rated by three people. On the training set, one of the three ratings was provided by the author of this work.[16] For evaluations on the testing set, all three ratings were from outside raters. There were eight outside raters, all of whom were university students and self-identified native speakers of English. One was an NLP expert but was not otherwise involved in this work.[17]

4.4.2 Mapping Ratings to Acceptability Labels

We mapped the ratings from individual raters to values from 1 to 5 (where 1 corresponds to "bad," 2 to "unacceptable," etc.), and then used the mean of the three ratings for a question as that question's 1 to 5 acceptability label (note that this can be fractional).

Also, in later sections, we use a threshold of 3.5 to induce boolean distinctions for question acceptability. We use the term "acceptable" to refer to labeled questions with mean ratings above the 3.5 threshold. Note that this threshold roughly corresponds to whether or not a majority of the raters judged a question to be acceptable or good. Using a five-point scale for ratings and then a boolean distinction for evaluation purposes has the following advantages:

- The five-point rating task is somewhat easier for individual annotators since they can select the "borderline" category if they are not sure about a rating.
- The binary conversion simplifies our analyses, enabling the use of standard evaluation metrics for binary predictions.

[16] The mean of the training set ratings from the author, which was 2.54 (Std. Dev. = 1.21), was close to the mean of all the training set ratings from the other raters, which was 2.43 (Std. Dev. = 1.35).

[17] Heilman and Smith (2010c) ran a pilot study to test the viability of crowdsourcing for rating computer-generated questions, using Amazon Mechanical Turk. The results were promising, but we chose not to use crowdsourcing technology here since it is still relatively new and not well understood.

Input Sentence	Question	Rating
In 1924 the site was chosen to serve as the capital of the new Tajik Autonomous S. S. R. . . . , and rapid industrial and population growth followed.	What followed?	1.00
Parliament offered the Crown not to James's eldest son . . . but to William and Mary as joint Sovereigns.	Who did Parliament offer the Crown not to?	2.00
The National Archives has exhibits of historical documents.	What does the National Archives have exhibits of?	3.00
After the People's Republic of China took control of Tibet in 1950 and suppressed a Tibetan uprising in 1959, the passes into Sikkim became a conduit for refugees from Tibet.	What did the People's Republic of China take control of in 1950?	3.67
Asmara was a village of the Tigre people until the late 19th century.	What was Asmara until the late 19th century?	4.00
Each year the banquet for the winners of the Nobel Prizes is held in City Hall.	Where is the banquet for the winners of the Nobel Prizes held?	4.67

Table 4.6: Examples of questions with their mean ratings, which range from 1 to 5.

- The finer-grained 1 to 5 ratings may be useful for training the statistical ranking model (though we do not test this directly).

Table 4.6 shows some examples of the mean ratings given to questions (from the training set used later in §4.7).

4.4.3 Inter-rater Agreement

To measure inter-rater agreement, we computed two agreement statistics. First, we calculated the percentage of questions for which there was unanimous agreement among the three raters about whether the question was "acceptable" or better versus "borderline" or worse (i.e., whether a rating was above or below 3.5). Second, we computed Fleiss's κ for this boolean distinction. Fleiss's κ is an agreement statistic for multiple raters that adjusts for chance agreement and ranges from zero to one.

For ratings of questions from the training set of rated questions described in §4.7, there was unanimous agreement for 59% of questions and a Fleiss's κ value of 0.34. For experiments with the testing set, we were somewhat more careful in selecting the outside raters and providing them with guidance (e.g., after they rated questions for the practice article). For the ratings of questions for the

document-level evaluation on the testing set described later in §4.10.1, there was unanimous agreement for 63% of questions and a Fleiss's κ of 0.51. From the ratings for the sentence-level evaluation described in §4.10.2, there was unanimous agreement for 73% of questions and a Fleiss's κ of 0.58. The κ values for the testing set correspond to "moderate agreement" (Landis and Koch, 1977) but are somewhat lower than agreement values reported for some other NLP problems.[18] Thus, while our rating scheme is less complex than the scheme from Heilman and Smith (2009), it also does not lead to extremely high agreement.

4.5 Samples of Texts and Questions

This section describes the two distinct samples of texts that are used in subsequent experiments.

The first sample, which we refer to as the **training set**, consists of 40 texts selected from the Wikipedia, Encyclopedia Britannica, and Britannica Elementary corpora using stratified random sampling. (These text sources are described in more detail §4.1.)

From these 40 texts, we created a set of 800 labeled questions for the ranker as follows. First, using stages 1 and 2 of our system (i.e., without ranking), we generated WH questions about the articles. We then randomly sampled 20 questions per article and obtained acceptability labels using human ratings, as described in §4.4. Note that we restrict our investigation to WH questions. That is, we exclude yes-no questions, which our system is capable of generating (§3.3 of Chapter 3) but which are easier to generate and probably less interesting, both in terms of their value as assessments and the range of linguistic phenomena they exhibit.

We use the training set for the following purposes.

- First, by training on a portion of these texts and testing on the rest, we evaluate whether the question ranker improves the quality of top-ranked questions (§4.7).
- Second, we use all 800 questions to perform some exploratory analyses of the question ranking model and its performance (§4.8).

[18] E.g., Dolan et al. (2004) and Glickman et al. (2005) report Cohen's κ values of around 0.6 for paraphrase identification and recognizing textual entailment, respectively. Cohen's κ is an agreement measure for two raters that is roughly comparable to Fleiss's κ. The guidelines of Landis and Koch (1977) apply to both Cohen's κ for two raters and Fleiss's κ for multiple rates.

	Training			Testing		
Source	**Texts**	**Mean Sent.**	**Mean Words**	**Texts**	**Mean Sent.**	**Mean Words**
Wikipedia	20	66	1,400	15	76	1,500
Encyclopedia Britannica	10	68	1,500	8	70	1,500
Britannica Elementary	10	42	570	7	35	480
All	40	60	1,200	30	65	1,300

Table 4.7: Descriptive statistics about the sets of articles used for training and testing: numbers of articles, the mean numbers of sentences per article, and the mean numbers of words per article. For clarity, mean values are given to two significant digits.

- Third, we use all 800 questions to train the ranker for subsequent experiments (§4.10, §4.11, and Chapter 5).

The second sample of texts, which we refer to as the held-out **testing set**, is used to study the performance of the end-to-end system in a more realistic ranking scenario. Rather than sampling from the unranked questions for each text, we will consider only the top few ranked questions. These questions that appear at the very top of the ranked lists are arguably the most relevant for potential applications because a user interface, such as the one we describe in Chapter 5, would present them first. The evaluations on this set of texts are described in more detail in §4.10.

Table 4.7 shows the numbers of texts in each set as well as the mean numbers of sentences per text and words per text. The Wikipedia and Britannica articles are roughly comparable in length, both being about 6 standard book pages on average (at 250 words per page). The Britannica Elementary articles are much shorter, about 2 book pages in length. The sentences in the Britannica Elementary texts are also shorter. The mean number of words per sentence for Britannica Elementary articles in the training set is 13.3, compared to 21.1 for Britannica and 21.2 for Wikipedia.

4.6 Evaluation Metrics

For measuring the quality of the output of the system, we compute the mean percentage of top-ranked questions that are rated acceptable. We use two variations of this metric.

In §4.7, to measure performance for a text in the training set, we estimate the percentage of questions that were rated acceptable out of the top ranked 20% of the sampled questions for the text. We

use the term **precision-at-20%** to refer to this metric. Recall that for each training set text, we have 20 questions sampled from the set of *all* questions. As such, this metric is computed from the top 4 questions.

Later, in §4.10, to measure performance on testing set texts, we acquire acceptability ratings for the top ranked N questions, where $N \leq 10$. We then compute the percentage of those questions that are acceptable. We use the term **precision-at-N** to refer to this metric.

Both of these metrics encode the assumption that users are likely to focus mainly on the first questions that a QG system presents (i.e., the top ranked ones).

Note that precision-at-N cannot be computed for the training set texts since we only have ratings for a sample of unranked questions generated from them. Also, precision-at-20% cannot be computed for test set texts since we only acquire ratings for the top N ranked questions.

In preliminary experiments, we also considered alternative ranking metrics such as Kendall's τ and mean average precision, but we found these other metrics did not seem to provide any additional information over the metrics discussed above. Also, we found them more difficult to interpret than precision-at-20% and precision-at-N.

4.7 Question Ranking Evaluation

In this section, we use the training set to study the effects of question ranking on the acceptability of the top-ranked questions at the document-level. Heilman and Smith (2010b) previously found that a statistical ranking model can roughly double the quality of top-ranked questions produced by a QG system. However, they ranked questions at the corpus-level rather than the document-level, combining questions from multiple texts into a single ranking. Here, we replicate the main finding from Heilman and Smith (2010b) that ranking improves QG performance—but at the level of the document.

We also explore whether other types of rankers could improve performance compared to the penalized linear regression approach that we employ. We intend to show whether or not the linear regression model performs well compared to other reasonable alternatives.

For the experiments in this section, we randomly split the training set into equal-sized parts.

Each consists of 20 texts and the 400 labeled questions from those texts. We used one half to tune hyperparameters for the ranking models, using leave-one-out cross-validation.[19] We then trained a ranking model on this first half and used it to rank questions from the second, held-out half. Our evaluation metric is precision-at-20%.

4.7.1 Passive Aggressive Learning Algorithms for Ranking

We compare the performance of the linear regression ranker to three maximum margin-based models trained using online passive aggressive (PA) learning (Crammer et al., 2006). PA learning and the closely related MIRA algorithm (Crammer and Singer, 2003) have been shown to lead to excellent performance on other NLP tasks (McDonald et al., 2005; Chiang et al., 2008).

In this section, we briefly describe the three alternative ranking methods used in our experiments. Appendix D presents them in more detail. We describe a PA version of linear regression for ranking, a ranker based on binary classification, and a ranker for pairwise preferences. These rankers can be viewed as instances of a general framework for PA learning, based on work by (Crammer et al., 2006), which we discuss in this section. The rankers are distinguished by particular representations, loss functions, and prediction rules.

The first method (PA-regression) is an online PA version of linear regression similar to support vector regression (Smola and Schölkopf, 2004). This regression model is adapted from Crammer et al. (2006). Like least squares linear regression, this model assumes that the data are on an interval scale of measurement (i.e., that there is a linear relationship between the feature vector and the output variable). Then, as with the least squares regression model discussed in §3.4.1 of Chapter 3, we rank questions by sorting by their predicted scores.

The second method (PA-binary) ranks questions by sorting the scores from a binary classifier for whether a question is acceptable or not. This binary classification approach is adapted from Crammer et al. (2006). Like logistic regression, it assumes a discrete, binary scale of measurement for

[19] To tune with leave-one-out cross-validation, we iteratively held out each article, trained the ranker using the other $N - 1$ articles, and measured precision-at-20% for the held out article. For a given setting of the hyperparameters, we can thus compute an average precision-at-20%. We selected the setting that led to the highest precision-at-20%. For the PA rankers, we tried the following hyperparameter values: $T \in \{1, 5, 10, 50, 100, 500, 1000\}$ and $C \in \{0.1, 1, 10\}$. For PA regression, we fixed $\epsilon = 0.001$. See Appendix D for details about the hyperparameters for the PA rankers. For the linear regression model (§3.4.1 in Chapter 3), we tried $\lambda \in \{0.1, 0.5, 1, 5, 10, 50, 100, 500, 1000, 5000\}$.

question quality. That is, the output can take two values: $\mathcal{Y} = \{-1, +1\}$, where $y = -1$ denotes an unacceptable question and $y = 1$ denotes an acceptable question. To train the model, we map 1 to 5 question acceptability ratings to binary values that indicate whether the threshold of 3.5 is exceeded. To rank new questions, we simply sort by their predicted real-valued scores for the "acceptable" class.

The third method (PA-pairwise) is a pairwise ranking model, which assumes that there is a natural ordering in the data but does not assume an interval scale of measurement. Pairwise ranking models, such as SVM$_{rank}$ (Joachims, 2002) and the ranking perceptron (Gao et al., 2005; Shen and Joshi, 2005), have been popular for creating ranked lists of retrieved documents in information retrieval applications. This method is similar the ranking perceptron, except that it uses PA learning instead of the perceptron learning algorithm (a relatively minor difference).

4.7.2 Results

Table 4.8 shows the results of ranking with each method. It also shows, in the "No ranking" row, the percentage of all questions in the held-out half of the training set that were rated acceptable (23%), which provides an estimate of performance without ranking (i.e., a baseline).

Ranking significantly increased the acceptability of the top-ranked questions. Precision-at-20% for the linear regression ranker used in the QG system was 45%, close to double the overall acceptability rate for the held-out half (23%). This increase is statistically significant (Wilcoxon signed rank test, $p < 0.001$).

The precision-at-20% values for the other ranking methods are relatively close, though somewhat lower. For example, precision-at-20% for the PA-regression ranker was 43%. However, probably due to the relatively small number of texts used for this experiment, none of the differences between the linear regression ranker and the others are statistically significant.[20]

In general, it seems that for this problem, with this data set, and with these features, the penalized least squares linear regression ranker performs about as well as the other techniques. It is possible that some of the recent work on learning to rank, particularly listwise ranking models that directly

[20]For the difference between the linear regression and PA-regression, $p = 0.727$ (Wilcoxon signed rank test). For the difference between the linear regression and PA-pairwise, $p = 0.057$ (Wilcoxon signed rank test). For the difference between the linear regression and PA-binary, $p = 0.119$ (Wilcoxon signed rank test).

Ranker	Precision-at-20% Mean	Std. Dev.
Linear Regression	45%	22%
PA-regression	43%	18%
PA-binary	36%	23%
PA-pairwise	35%	24%
No ranking	23%	8%

Table 4.8: Performance for various ranking methods. Results are averaged across the texts in the held-out half of the training set. The "No ranking" row shows the overall acceptability rate for all questions in the held-out half.

optimize evaluation metrics (Chakrabarti et al., 2008; Xu and Li, 2007), could lead to improved ranking performance.[21] We leave the exploration of such models to future work.

4.8 Further Analyses of Question Ranking

Next, we discuss some exploratory analyses conducted to better understand certain aspects of question ranking performance. For these analyses, we use the whole training set of 40 articles and 800 questions. In §4.8.1, §4.8.2, and and §4.8.4, we use leave-one-out cross-validation to evaluate the performance of the ranker.[22] Again, our evaluation metric is precision-at-20%.

4.8.1 Effect of Training Set Size

To test the effect of the size of the training data, we used leave-one-out cross-validation with varying numbers of texts used for training. For each held-out article, we varied the number of articles M used to train the ranker from 1 to $N - 1$. For each M, we performed ten replications of randomly sampling (without replacement) from the $N - 1$ articles, and computed precision-at-20%. We then combined the precision-at-20% values for each held-out article in order to compute means and stan-

[21] See `http://research.microsoft.com/en-us/um/people/letor/` for more information about the learning to rank (LETOR) task.

[22] For these analyses, the end-to-end evaluations (§4.10), and the user study (Chapter 5), we intended to choose the ranker's λ hyperparameter to maximize precision-at-20% in leave-one-out cross-validation on the training set. However, we accidentally selected a suboptimal value: we used $\lambda = 500$, which led to a precision-at-20% of 36.9%, whereas $\lambda = 1$ and $\lambda = 0.5$ led to 37.5%. Values in between led to performance that was equal to or slightly worse than 36.9%. In general, performance does not seem to strongly depend on the particular setting of λ; a wide range of settings led to similar performance. Note that this issue may make the evaluations slightly pessimistic.

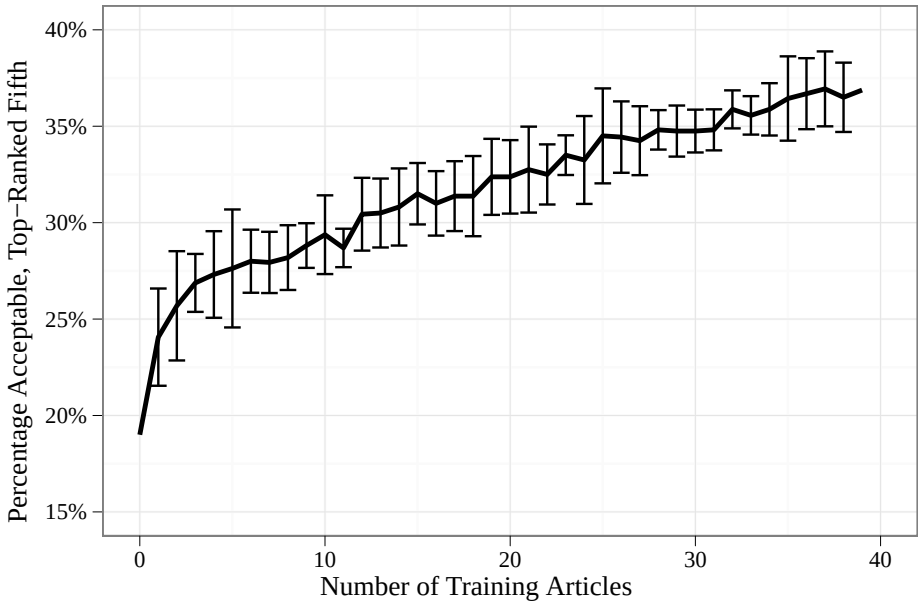

Figure 4.1: The percentage of acceptable questions in the top-ranked fifth (i.e., precision-at-20%) as the size of the ranker's training set increases, estimated using leave-one-out cross-validation (see text for details). The solid line and error bars indicate the means and standard deviations across ten replications of randomly sampling subsets of articles.

dard deviations for each M. As shown in Figure 4.1, when using $N - 1$ articles to train the ranker, the precision-at-20% is 37%. This acceptability rate is roughly double the acceptability rate for the entire training set, which was 19%. Most of the gain occurs with a small amount of training data: the mean percent acceptability increases from 19% to 28% with 5 training articles (100 questions), and then to 32% with 20 articles (400 questions).

4.8.2 Distribution of Question Ratings with and without Ranking

We also created histograms of the 1 to 5 question rating labels before and after ranking. To measure the distribution before ranking, we used all 800 labeled questions from the training set. To measure the distribution after ranking, we selected just the top-ranked fifth of questions for each of the training set articles in the rankings produced from training on the other $N - 1$ articles. Figure 4.2 shows the resulting histograms. Ranking substantially shifted the distribution of question quality to the

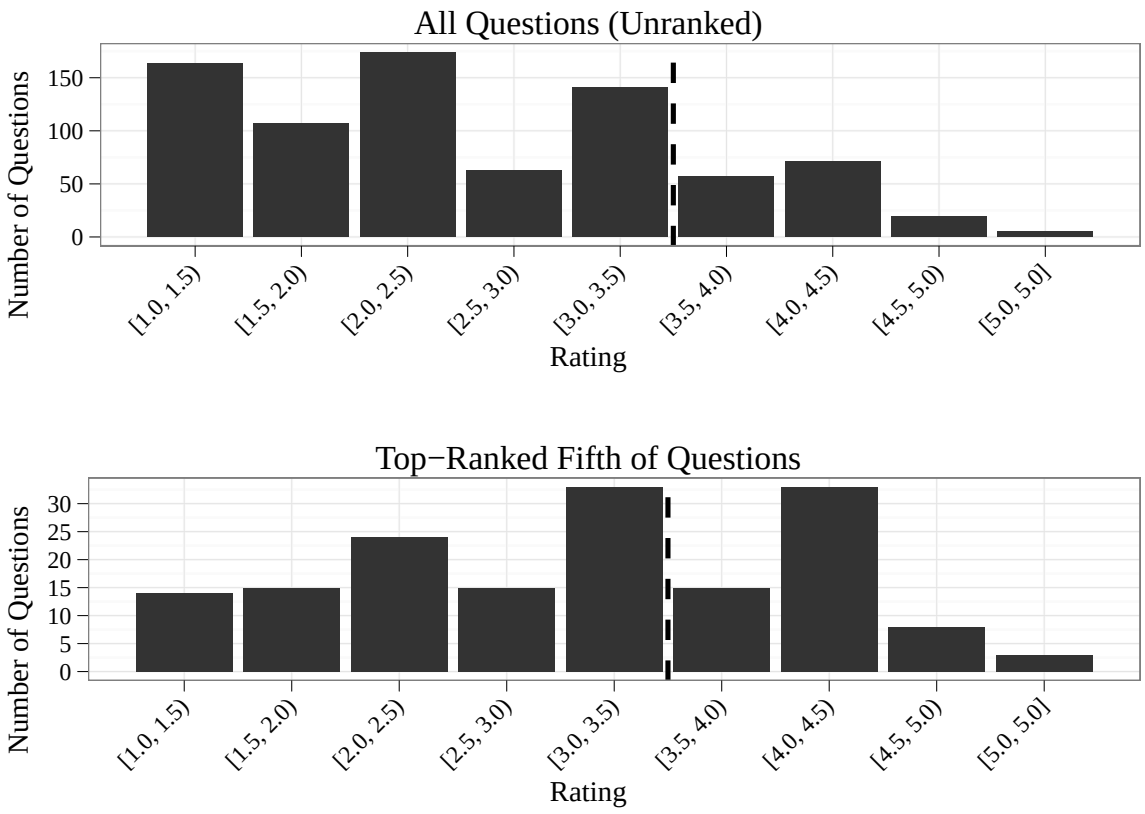

Figure 4.2: Distributions of ratings for questions, before and after ranking. The top graph shows the distribution ratings for all questions (without ranking), and the bottom graph shows the distribution of the top-ranked fifth of questions, using leave-one-out cross-validation to evaluate the ranker. The vertical dashed lines indicate the threshold for a question being "acceptable" (i.e., whether its mean rating is above or below 3.5).

right—toward more higher quality questions. Also, a substantial fraction of the top-ranked questions ($\sim$20%) had ratings between 3.0 and 3.5, indicating that they are "borderline" questions—some of which might easily be revised by a human user. Note that the percentage of ranked questions above the 3.5 threshold in the figure is the precision-at-20%.

4.8.3 Analysis of Feature Weights

In this section, we explore the feature weight parameters learned by the ranker. Table 4.9 shows some of the feature weights for the question ranker trained on the whole training set of 800 questions.[23] See §3.4.2 of Chapter 3 for full descriptions of the features. For conciseness, Table 4.9 does not include values for all of the features in the model. Note that this model is the one used in subsequent evaluations of the end-to-end system (in §4.10 and Chapter 5).

The ranker induces a preference for the different possible WH words. For example, *who* questions, whose answers are often the names of specific individuals, receive a positive weight, while *what* questions, whose answers cover a wider range of phrases, receive a negative weight.

The ranker also learns to slightly disprefer most of the operations that are performed by the simplified statement extractor in stage 1 (§3.2.1 of Chapter 3), assigning them negative weights. Not surprisingly, these operations appear to be somewhat more likely to lead to bad questions, probably because they cause the system to rely on potentially inaccurate syntactic parses to a greater extent than when the system just asks questions about the main clause. Similarly, the use of pronoun resolution receives a negative weight, probably because it causes the system to rely on potentially inaccurate coreference resolutions. Of course, these weights are not so strong as to make the system completely avoid questions with these features.

Looking at features for particular parts of speech, we observe that higher numbers of proper nouns and noun phrases are associated with good questions, perhaps because more nouns indicate a higher level of specificity in the output. In contrast, the "vague noun phrase" feature receives a negative weight, as do the features for the number of adverbs (which may lead to awkward questions such as *Who did John then meet?*).

4.8.4 Feature Ablation Study

We performed ablation experiments to study the effects of removing each of the different types of features described in §3.4.2 of Chapter 3. We used leave-one-article-out cross-validation on the full training set, as in §4.8.1. We trained a linear regression ranker for each set of features that results

[23] Training the ranker on subsets of the data produces weight vectors that appear broadly but not completely similar. Many important features receive weights of the same sign and comparable magnitude.

Feature Description	Value
Question contained *who*	0.367
Question contained *what*	-0.049
Question contained *where*	0.260
Question contained *when*	0.103
Question contained *how many*	0.065
Question contained *whose*	-0.032
Answer phrase was the subject	0.112
Main verb was a form of *be*	0.028
Extracted from an appositive	-0.210
Extracted from a temporal subordinate clause	-0.173
Extracted from a participial phrase	-0.085
Extracted from a relative clause	0.049
Extracted from a conjunction	0.005
Pronoun resolution	-0.104
Number of proper nouns in question	0.018
Number of proper nouns in answer	0.013
Number of noun phrases in question	0.027
Number of noun phrases in answer	0.013
Number of "vague noun phrases" in question	-0.015
Number of "vague noun phrases" in answer	-0.020
Number of adverbs in question	-0.061
Number of adverbs in answer	-0.108

Table 4.9: Some of the interpretable and interesting feature weight values learned by the question ranker.

from removing a single feature type from the full set. For each of these subsets, we measured the percentage of acceptable questions in the top-ranked 20% averaged across all of the training articles (each of which is held out during cross-validation).

Table 4.10 presents the results, along with the overall percentage of (unranked) acceptable questions. The grammatical features (e.g., counts of different syntactic categories) appear to be the most important: removing them from the feature set resulted in a 4% absolute drop in acceptability of the top 20% of questions, from 37% to 33%. Transformation features also appear to be important because taking them out led to a 3% drop in performance. In contrast, some of the features did not appear to be particularly helpful. For example, removing WH and length features actually increased the precision-at-20% very slightly, though this may be due to variance in our relatively small sample of 40 articles.

A similar ablation experiment was reported by Heilman and Smith (2010b). They used an earlier

Features	Precision-at-20% Mean	Std. Dev.
All	37%	28%
All – Length	38%	28%
All – WH	38%	28%
All – Negation	37%	28%
All – LangModel	36%	27%
All – Grammatical	33%	23%
All – Transforms	34%	26%
All – Vagueness	35%	28%
All – Histograms	34%	23%
All – Pronoun Repl.	37%	28%
"No Ranking"	19%	10%

Table 4.10: The percentages of the top ranked 20% of questions that were labeled acceptable, for rankers built from variations of the complete set of features ("All"). E.g., "All – WH" is the set of all features *except* WH word features. The "No ranking" row shows the overall acceptability rate for all questions in the training set.

version of the QG system and included yes-no questions in addition to WH questions, but found similar results—in particular, that the grammatical features were the most important. They found at least small advantages for all but the length features.

4.9 Effects of WH-movement Constraints

We conducted a small-scale study of the quality and quantity of the questions that the system avoids by constraining WH-movement (i.e., by identifying phrases that would lead to ungrammatical questions if they were to undergo WH-movement, as described in §3.3.1 of Chapter 3).

For this study, we disabled the WH constraints in the system and generated questions from the 40 articles in the training set. Without the WH constraints, the system allows every noun phrase, prepositional phrase, and subordinate clause (i.e., node with the label "SBAR") in declarative sentences to be an answer phrase. A total of 15,876 questions were generated, which more than doubles the 7,417 generated by the system when the constraints are active. For each article, we then randomly sampled two questions from the set of those that would have been avoided by the full system due to the WH constraints. The author of this work rated each sampled question for acceptability using the 1 to 5

scale described in §4.4.

The great majority of the questions were ungrammatical and nonsensical. Of the 80 sampled questions, 69 (86%) were rated 2 ("unacceptable") or below. For example, from the sentence 4.6, the ungrammatical question 4.7 was generated.

(4.6) In the 6th century a group known as the Slovenes began settling at the site and built a new town.

(4.7) $*$ Who did a group known as begin settling at the site?

The outputs from the full system—question 4.8, for example—are not much better in this case.

(4.8) $\otimes$ What began settling at the site?

However, at least question 4.8 is grammatical, even though it lacks temporal context and contains the vague noun phrase *the site*.

Nine of the sampled questions (11%) were rated as borderline. A few of these were potentially acceptable but rather awkwardly phrased. For example, from sentence 4.9, question 4.10 was generated by selecting the deeply embedded noun phrase *Eritrea* as an answer phrase.

(4.9) Asmara is the capital and largest city of the East African country of Eritrea.

(4.10) $\otimes$ What is Asmara the capital and largest city of the East African country of?

Only two of the sampled questions were judged acceptable (i.e., only 3% of the sample). For example, from sentence 4.11, where *the country* refers to South Korea, question 4.12 was generated.

(4.11) Seoul is the center of finance for the country.

(4.12) What is Seoul the center of finance for?

The parse tree for sentence 4.11 has the entire sequence of words *the center of finance for the country* as a noun phrase. The full system does not generate question 4.12 because it treats this noun phrase as an island—and therefore does not extract the subtree *for the country* as a potential answer

phrase. We note that the full system would have generated question 4.12 if the phrase *for the country* had been attached to the main verb *is* rather than the noun *center*.

In contrast to the two acceptable questions (i.e., 3%) generated without movements constraints, recall from §4.8 that 19% of the unranked questions generated by running the system on the training set were rated as acceptable. Therefore, it appears the acceptability rate for questions filtered out by the WH constraints is much lower than for the questions the full system generates.

It is difficult to test whether the system's WH-movement constraints are overly permissive—that is, whether they allow answer phrases to undergo WH-movement when they should not. However, in our analyses of the errors made by the system (discussed later in §4.11), we did not observe any errors due to missing syntactic constraints. We did observe a number of unacceptable questions in which the answer phrase was part of an idiomatic multi-word expression (e.g., *the ground* in *burned to the ground*), but such phenomena are arguably more lexical than syntactic in nature.

It is worth noting that the full set of constraints is not necessary to generate many of the acceptable questions from the full system. Simpler, more aggressive constraints would still likely yield many acceptable questions. One could, for instance, only allow noun phrases in the subject position to be answer phrases. Forty-five percent of the training set questions that were labeled as acceptable could still be generated with just this simple, aggressive constraint in place. Therefore, while the set of movement constraints appears to be effective at filtering out ungrammatical questions, it may be somewhat more complex than necessary if relatively few acceptable questions are needed for a particular application.

4.10 End-to-end Evaluation Experiments

In this section, we discuss experiments that involve the full overgenerate-and-rank QG system. Our experimental designs and evaluation metrics are designed around two ways in which candidate questions in a QG system could be presented to a user via user interface (note that the user interface described in Chapter 5 is designed to support these two scenarios).

- **Document-level**: First, a user might wish to view a list of questions for the entire text, ranked by the QG system's predictions of quality.

- **Sentence-level**: Second, a user might wish to view a list of questions on a sentence-by-sentence basis—for example, by mousing over or clicking on parts of the text.

In both of these scenarios, a user is unlikely to consider more than a few questions suggested by the system. For that reason, these evaluations measure performance with only the top few most highly ranked questions for a document or a sentence, rather than with samples of unranked questions as in the experiments with the training set discussed in previous sections.

Here, we ask the following high-level research questions:

- What is the overall level of acceptability, according to general linguistic factors such as grammaticality, of the top-ranked questions generated by our QG system?
- How does performance compare at the document-level and the sentence-level?
- What are the characteristics (e.g., WH words used, transformations applied) of the acceptable top-ranked questions at the document-level and sentence level?

4.10.1 Document-Level Evaluation

This section describes experiments that evaluate the quality of the output of the QG system at the document level. We evaluate the QG system by measuring the acceptability rates of the top-ranked questions generated about held-out testing set articles (§4.5).

We generated questions and selected the top-ranked 10 questions for each testing set article, using the ranker trained on the full training set of 800 questions. The top questions were then rated by human annotators (as discussed in §4.4). With these ratings, we measured precision-at-N, the mean percentage of questions of the top N questions that were rated acceptable, as discussed in §4.6.

Averaging across the articles in the testing set, approximately two fifths of the top-ranked questions were rated acceptable: the mean precision-at-10 was 42% (Std. Dev. = 19%). As shown in Table 4.11, precision-at-N values for smaller N were similar. For example, precision-at-5 was 49% (Std. Dev. = 26%). Note that these acceptability rates are much higher than the average acceptability rate for all of the unranked questions from the training set, which was only 19% (Std. Dev. =

Rank N	Precision-at-N Mean	Precision-at-N Std. Dev.	Num. Acceptable Per Document Mean	Num. Acceptable Per Document Std. Dev.
1	47%	51%	0.5	0.5
2	38%	36%	0.8	0.7
3	46%	32%	1.4	1.0
4	44%	29%	1.8	1.2
5	49%	26%	2.4	1.3
6	48%	24%	2.9	1.4
7	46%	23%	3.2	1.6
8	45%	22%	3.6	1.7
9	44%	19%	4.0	1.7
10	42%	19%	4.2	1.9

Table 4.11: Document-level performance for different values of the maximum rank N (i.e., the number of top ranked questions to consider per article). In addition to precision-at-N, the table includes the average number of acceptable questions per article up to rank N.

10%).[24]

Table 4.11 also reports means and standard deviations for the total numbers of acceptable questions up to rank N. In the top 3 questions for a text, there were 1.4 acceptable questions, on average. In the top 10, there were about 4.2, on average. These values suggest that in order to find a few acceptable questions for a text, a user would probably not have to look very far down the ranked list.

As indicated by the high standard deviation values for precision-at-N, the quality of the output varied considerably by the topic of the input article. For example, for the Wikipedia article about *Banksia epica* (a shrub that grows in Western Australia), only two of the top ten questions were rated acceptable. One reason for the low performance on that article appears to be that its language is very technical (e.g., *Like most other Proteaceae, B. epica has proteoid roots, roots with dense clusters of short lateral rootlets that form a mat in the soil just below the leaf litter*). Another reason is that the supersense tagger mislabeled instances of *Banksia epica* and *B. epica* as `noun.person`, leading to many erroneous *who* questions.

In contrast, for the Britannica Elementary article about Monrovia (the capital of Liberia), five of the top ten questions were rated acceptable, as shown in Table 4.12.

[24] It is not straightforward to perform a test of statistical significance comparing the training and testing set acceptability rates. Since the ranker was trained using the training set, questions from the training and testing sets are not independent, even though the two sets of articles were drawn independently from the same sources. Recall, however, that the experiments in §4.7 already demonstrated that ranking significantly improves acceptability.

Rank	Question	Answer Phrase	Avg. Rating
1	Who was president of the United States in 1822?	James Monroe	5.00
2	Who was James Monroe president of in 1822?	the United States	2.00
3	When did Monrovia become the capital of the new country of Liberia?	in 1847	4.67
4	Who was James Monroe in 1822?	president of the United States	1.67
5	What has a large port on the Atlantic Ocean that handles such products as rubber, timber, and iron ore?	Monrovia	4.33
6	What is capital of the western African country of Liberia?	Monrovia	4.67
7	When was James Monroe president of the United States?	in 1822	4.67
8	What did the Organization of African Unity emerge from in 1961?	from a meeting in Monrovia	3.00
9	What left the country in the periods of fighting?	many other people	1.00
10	What emerged from a meeting in Monrovia in 1961?	The Organization of African Unity	3.33

Table 4.12: The top ten questions for the Britannica Elementary article about the city of Monrovia, with their answers and average acceptability ratings.

Many of the questions in Table 4.12 cover similar information about Monrovia. Whether or not it is useful in general to show similar questions to users is unclear, but it seems that in some applications, a diverse set of questions would be desired. Future work could explore using information retrieval techniques such as maximal marginal relevance (Carbonell and Goldstein, 1998) to promote diversity in the output. Or, a user interface for QG could hide questions that are related to one that a user just selected.

More generally, the performance of the system varies considerably by the source of the article, as shown in Figure 4.3. Interestingly, the system performs best on Britannica Elementary articles, where it achieves 56% precision-at-10, compared to 39% for Wikipedia articles and 36% for Britannica articles. The difference between the precision-at-10 for Britannica Elementary texts and the precision-at-10 for all texts from the other sources is statistically significant (Wilcoxon rank-sum test, $p = 0.024$).

We attribute the difference in performance to the fact that the Britannica Elementary articles are written with fewer complex sentences in order to maintain a low reading difficulty level. This re-

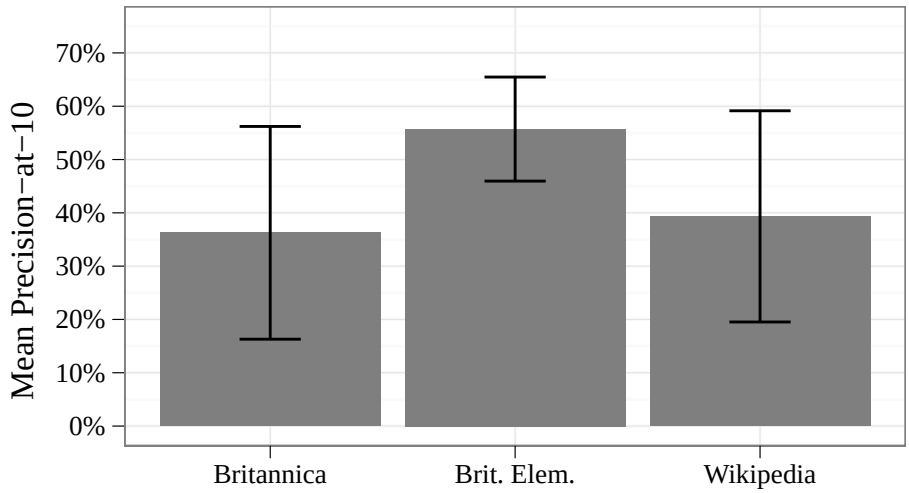

Figure 4.3: Document-level precision-at-10 values for different sources of texts. Error bars indicate standard deviations.

duction in linguistic complexity makes it more likely that the pre-processing tools in our system will correctly analyze the structures of the input sentences (§4.11 discusses how pre-processing is a source of errors). A reduction in preprocessing errors seems particularly likely for the parser, since syntactic parsers are commonly found to perform better on shorter sentences (Collins, 1999; Charniak, 2000). Also, it is worth noting that these Britannica Elementary articles are the closest in terms of reading level and style to the sort of texts that an elementary school student might encounter in a classroom.

One issue that we leave to future work is how ranking performance—or the system's performance more generally—would be affected by changing to a new text domain. For example, a ranker trained on informational texts might not perform as well at ranking questions generated from narrative fiction texts. Various extensions to the ranker's feature set might be needed to model linguistic phenomena that occur more frequently in a new domain (e.g., quotations and metaphorical language tend to occur frequently in narratives).

4.10.2 Sentence-Level Evaluation

Next, we evaluate the QG system at the level of individual sentences in order to answer the following research questions:

- What is the rate of question acceptability for the system at the sentence level?
- How many acceptable questions are there in the top N for a single sentence, on average?

The latter question is motivated by the assumption that a person using a QG tool could quickly consider a small number of suggested questions per sentence (e.g., N up to 3). So, even if the rate of acceptability (as measured by precision-at-N) is far from 100%, a user may still benefit from sentence-level suggestions if one or more acceptable questions can be suggested, on average.

We begin with some descriptive statistics. There are 1,947 sentences in the testing set, including titles and section headings, and the system generated a total of 5,413 questions (2.8 per sentence). The system generated at least one question for 1,482 sentences (76% of 1,947).

To evaluate the top-ranked questions, we randomly sampled, for each of the thirty articles in the testing set, two sentences from the set of sentences in the article where at least one question was generated. Then, for each sentence, we gathered human judgments for the acceptability of the top three questions. Note that the system produced fewer than three questions for some of the sentences. In this section, for simplicity, we micro-average acceptability rates at the question level rather than macro-averaging the sentence-level acceptability rates.

As shown in Table 4.13, precision-at-3 at the sentence level was 27%, which is around 20% lower than the 46% precision-at-3 we observed at the document-level (§4.10.1). This finding is not surprising since the ranker has fewer candidates to choose from when generating from a single sentence. Precision-at-1 at the sentence level (45%) was fairly close, however, to precision-at-1 at the document level (47%).

The system's list of top three ranked questions for a sentence included 0.75 acceptable questions on average. At lower ranks, where fewer questions are considered, the number of acceptable top-N questions decreased: At rank $N = 1$, there were 0.45 acceptable questions per sentence (since precision-at-1 was 45% and there was always at least one question per sentence in our sample). It is important to note that for many of the randomly sampled input sentences for this experiment, there

Rank N	Num. Acceptable	Num. Questions	Precision-at-N	Num. Acceptable Per Sentence
1	27	60	45%	0.45
2	39	119	33%	0.65
3	45	169	27%	0.75

Table 4.13: Numbers of acceptable questions, total numbers of questions, precision-at-N acceptability rates, and numbers of acceptable questions per sentence for different values of the maximum rank N (i.e., the number of top-ranked questions to consider per sentence).

may not be any acceptable factual questions (e.g., the input sentence *At this point no infrageneric classification had yet been suggested*).

As in the document-level evaluation, the source of the text affects the performance of the system, and the best performance is achieved on the relatively simple, well-edited sentences in the Britannica Elementary articles. While 27% of the top three sentence-level questions from all three sources were acceptable, 43% of those from Britannica Elementary sentences were acceptable. In contrast, 21% from Britannica sentences and 22% from Wikipedia sentences were acceptable.

4.10.3 Characteristics of Top-Ranked Questions

The QG system uses a variety of transformations, resulting in questions with different lengths, WH words, and other characteristics. In order to understand what types of questions were generated most often, we computed, for the document- and sentence-level evaluations, the percentages of top-ranked questions that use various WH words and transformations. We computed these percentages for all of the top-ranked questions and for just the acceptable ones.

Of the WH words, *what* questions were the most common, accounting for 50% of the acceptable top-ranked document-level questions. However, *who* and *when* questions were fairly frequent as well, each accounting for 21% of the acceptable outputs.

The simplified statement extraction operations accounted for a substantial fraction of the outputs, particularly at the sentence level. For example, extraction from conjunctions accounted for 31% of the acceptable questions at the sentence-level, and extraction from relative clauses accounted for around 9%.

Pronoun resolution (§3.2.2 of Chapter 3) was involved for a relatively small percentage of the

Question Type	Document-level Acceptable	Document-level All	Sentence-level Acceptable	Sentence-level All
Question was a *who* question	21%	22%	16%	18%
Question was a *what* question	50%	49%	58%	63%
Question was a *where* question	6%	9%	4%	4%
Question was a *when* question	21%	15%	16%	11%
Question was a *how many* question	2%	2%	4%	4%
Question was a *whose* question	1%	3%	2%	1%
Answer phrase was the subject	55%	48%	51%	44%
Main verb was a form of *be*	48%	48%	40%	39%
Extracted from an appositive	3%	3%	0%	5%
Extracted from a temporal subordinate clause	1%	1%	0%	3%
Extracted from a participial phrase	5%	4%	4%	4%
Extracted from a relative clause	11%	10%	9%	3%
Extracted from a conjunction	14%	15%	31%	29%
Pronoun resolution	7%	12%	13%	14%

Table 4.14: Percentages of acceptable top-ranked questions and all top-ranked questions at the document-level and sentence-level that had various WH words and transformations. The percentages of different question words do not add up to 100% due to rounding.

acceptable outputs: 7% at the document level and 13% at the sentence level.

More generally, the stage 1 transformations, both simplified statement extraction and pronoun resolution, appear to play a somewhat more important at the sentence level than the document level. For a whole document, one can generate many questions by performing basic transformations on the main clause. In contrast, there are relatively few questions one can generate from a single sentence without considering nested clauses and other complex constructions.

4.11 Error Analysis

This section presents the results of an error analysis we conducted to identify the major sources of errors in the QG system and to quantify the rates at which various types of errors occur. We analyzed a random sample of 100 of the unranked questions from the training set that were rated unacceptable (i.e., whose mean ratings were below 3.5), and 100 ranked questions from the document-level testing set evaluation that were rated unacceptable. In this way, since the training and testing sets were random samples from similar distributions, the effect of question ranking can be observed.

A few caveats are worth mentioning: First, the results in this section come from a relatively small

sample, so we hesitate to make strong conclusions. Second, the errors discussed here are only errors of *commission*—that is, errors that affect the precision of the system. They do not include errors of *omission*, where the system could have asked a question but did not—due, for example, to a lack of world knowledge or inference abilities, as discussed in §2.3.2 of Chapter 2. Such errors are more difficult to identify.

4.11.1 Error Types

We organize the errors made by the QG system into the following categories, most of which map onto challenges presented in Chapter 2.

- **Parsing errors:** Errors made by the Stanford Parser (Klein and Manning, 2003) can propagate and lead to errors, mainly in the simplified statement extraction and question creation steps (Chapter 3). See the discussion of this challenge in §2.2.1 of Chapter 2 for an example and further discussion.
- **Supersense tagging errors:** The supersense tagger (Ciaramita and Altun, 2006) often labels the head words of answer phrases with incorrect semantic types, as discussed in §2.1.1 of Chapter 2 and tested in §4.3. These incorrect labels can result in incorrect question phrases (e.g., *who* referring to a species of plant).
- **Pronoun resolution errors:** The pronoun resolution step sometimes replace pronouns with incorrect antecedent noun phrases, leading to questions that are incorrect or nonsensical. See §2.3.1 of Chapter 2 for examples and discussion.
- **Errors from taking information out of context:** As discussed in §2.3.1 of Chapter 2, questions can be vague when taken out of their original discourse context due to a lack of temporal or locational context (e.g., the question ⊗ *How many people were left homeless?* from the input sentence *Later,* [*Typhoon Paka*] *passed just north of Guam . . . 5,000 people were left homeless . . .*).
- **Vague noun phrases:** As discussed in §2.3.1 of Chapter 2, questions can also be under-specified if they contain noun phrases that refer to previously mentioned entities in the text (e.g., the phrase *the studio*, whose antecedent in the text is *Metro-Goldwyn-Mayer cartoon studio*, in the following question: ⊗ *What boosted the studio to the top of the TV cartoon field?*).

- **Bad question phrases:** As discussed in §2.1.1 of Chapter 2, question phrases can be unacceptable even if they use the correct WH word, since it is often the case that a more specific question phrase is needed (e.g., the question phrase *what illegal activity* would be better than *what* in the following question: ⊗ *What is a major concern of the Indian government?*, which was generated from the sentence *A major concern of the Indian government is the trafficking of wildlife products such as tiger and leopard skins . . .*). This error mostly occurs with *what* questions. Note this category does not include the errors caused by supersense tagging, which have their own category.
- **Decomposition errors:** As discussed in §2.1.3 of Chapter 2, the meaning of phrases and sentences is not always well modeled as a compositional process. The QG system makes the assumption of compositionality, however, leading to improper handling of multi-word expressions and other complex constructions. In addition to this general problem, there are a few kinds of errors that result from shortcomings in how stages 1 and 2 of the system handle certain types of constructions.
- **Formatting and dataset errors:** This relatively infrequent category consists of errors caused by various issues having to do with the format of the input texts. Occasionally, the input sentences themselves are ungrammatical or contain typographical errors. Also, the QG system sometimes generates questions from captions or titles if the parser does not identify them as sentence fragments (errors with captions, etc. could sometimes be categorized as parsing errors, but we chose to create a separate category since they might be addressed with simple filtering techniques).

Additional Examples

In order to supplement the discussion from Chapter 2 and to further illustrate some of the error types, this section provides a few additional examples of errors made by the QG system. These errors are taken from the sample of 200 questions described above.

The following example is a question that is syntactically well-formed but vague due to the extraction from its original context. The system takes the input sentence 4.13, which is from a Britannica Elementary article on the city of Manila, and is able to extract a simplified statement *The University of Santo Tomas was established in 1611.* However, the system converts this into question 4.14, whose question phrase *what* is probably not sufficiently specific.

(4.13) The University of Santo Tomas was established in 1611 and is the oldest university in the Far East.

(4.14) ⊗ What was established in 1611?

Some more appropriate question phrases might be, for example, *what university* or *what educational institution*. In addition, one might also consider the question to be lacking a phrase specifying the locational context (e.g., *in the city of Manila*), though we did not mark the question with this error type in our analysis because Manila is the subject of the article.

Next, we turn to a set of errors pertaining to the decomposition of multi-word phrases. Some errors are due to relatively simple issues that could be dealt with by extensions of the simplified statement extraction step. First, the system does not properly handle city-state combinations separated by commas since their syntactic structures look similar to subtrees for appositives (i.e., with two noun phrases offset by commas). For example, the system generates question 4.16 by removing what it thinks is an appositive (*Kentucky*) from the input sentence 4.15, which comes from the Wikipedia article on Richard Hawes, a Civil War era politician.

(4.15) Following the war, the Confederate government of Kentucky collapsed, and Hawes returned to his home in Paris, Kentucky.

(4.16) ⊗ Who returned to his home in Paris?

A better question that would not conflate Paris, Kentucky with the capital of France is *Who returned to his home in Paris, Kentucky?*. Such errors could be avoided by extending the system to consider semantic types when deciding whether to extract from appositives. Note that question 4.16 also lacks a specific temporal context (in this case, a modifier like *after the Civil War*).

The simplified statement extractor also occasionally makes errors with verb tenses. For example, from example 4.17, the system incorrectly uses the present tense form *are* of the copula *be* since the main clause is in present tense (cf. . . . *is a mainly sedentary form* . . .).

(4.17) T. p. hebridensis, described by British ornithologist William Eagle Clarke in 1913, is a mainly sedentary form found in the Outer Hebrides and Isle of Skye in Scotland.

From this sentence, the system extracts the statement in example 4.18, leading to the awkwardly phrased and unacceptable question 4.19.

(4.18) T. p. hebridensis are described by British ornithologist William Eagle Clarke in 1913.

(4.19) ⊗ What are described by British ornithologist William Eagle Clarke in 1913?

A number of decomposition errors occur as a result of particular syntactic constructions that span more than a single constituent in their respective parse trees. One common example is the *from X to Y* construction. Instead of selecting the pair of separate prepositional phrases *from September to February* in sentence 4.20, the system separately selects each prepositional phrase, producing questions 4.21 and 4.22.

(4.20) For Indian Ocean populations such as the Seychelles hawksbill population, the mating season is from September to February.

(4.21) ⊗ When is the mating season to February for Indian Ocean populations such as the Seychelles hawksbill population?

(4.22) ⊗ When is the mating season from September for Indian Ocean populations such as the Seychelles hawksbill population?

If a QG system could identify such constructions, it could generate better candidates such as the following.

(4.23) When is the mating season for Indian Ocean populations such as the Seychelles hawksbill population?

(4.24) When is the mating season for the Seychelles hawksbill population?

In addition, errors can result when separate syntactic constituents express related information, particularly temporal or locational information. Consider example 4.25, which is about Wellington, New Zealand.

(4.25) Rail and road services extend from it to all parts of North Island, and ferries to Picton link the capital to South Island.

The QG system selects *to South Island* as an answer phrase and generates question 4.26, which is nonsensical since it asks about a location when one is already specified in the question (*to Picton*).[25]

(4.26) ⊗ Where do ferries to Picton link the capital?

Since such examples exhibit linguistic constructions that are not well modeled by current syntactic parsers, it is not clear how to address them, except perhaps by producing specific rules for each possible construction (of which there may be many rare types). Future linguistic research may provide representations that facilitate the handling of these sorts of complex constructions.

4.11.2 Results

We categorized the 100 ranked and 100 unranked questions according to the error categories described above.[26] Each unacceptable question could belong to one or more categories, since it could exhibit multiple types of errors. The results are shown in Figure 4.4.

Many different types of errors appear in the samples of ranked and unranked questions. There is no single error type that dominates. However, some errors were more prevalent than others. Question phrase errors were particularly prevalent, appearing in 39% of unranked questions. Parsing errors were also common, appearing in 19% of unranked questions.

Preprocessing errors constituted a larger percentage of the errors when using the ranker. For example, parsing errors went from 19% of questions in the unranked training set to 30% of top-ranked questions from the testing set (note that this higher percentage does not mean that the overall rate of parsing errors increased since these are percentages of errorful questions, not of all questions). Similarly, supersense tagging errors showed a relative increase from 10% to 20%.

Apparently, while the question ranker can learn to avoid some of the other types of errors, it does not identify when the preprocessing tools make mistakes. This is not surprising since the preprocessing tools are complex models with a variety of features—and they are specifically designed and trained to avoid incorrect linguistic analyses, while the question ranker is not. Including confidence

[25] Observe that example 4.26 might be improved with a more specific WH phrase, such as *to which island*, and that it also contains a potentially vague noun phrase *the capital*.

[26] Since most of the error types require deep knowledge of how the system works (e.g., identifying when an error is due to the syntactic parser rather than some other component of the system), we did not check inter-rater agreement for this exploratory error analysis.

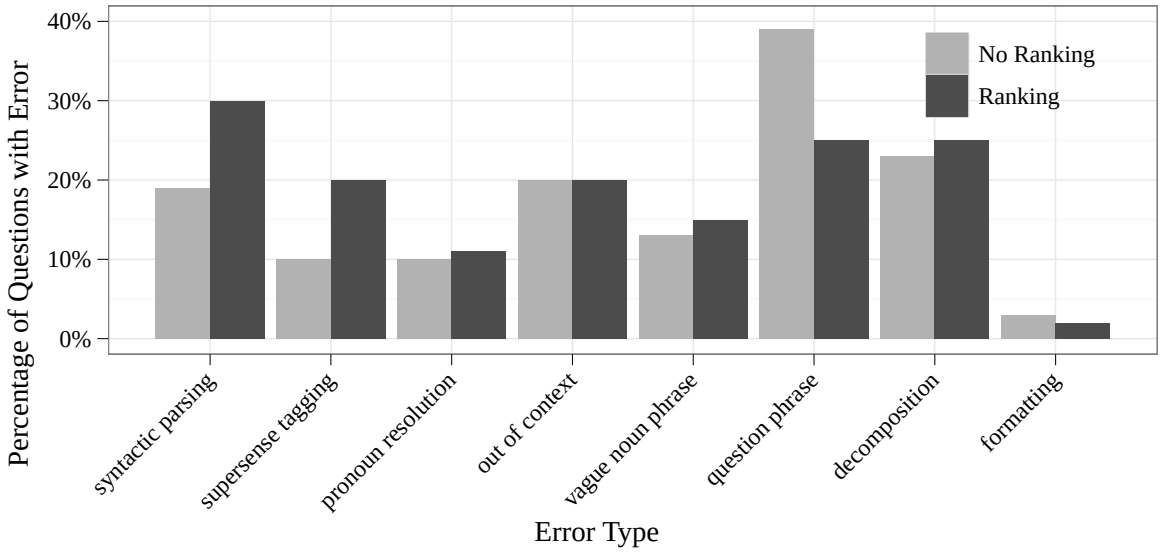

Figure 4.4: The rates of various error types in samples of top-ranked questions (from the document-level evaluation on testing set articles) and unranked questions (from the training set articles). Note that the values are percentages of *errors* not percentages of all questions, since both samples include only questions that were labeled unacceptable.

estimates as features might help the ranker avoid preprocessing errors; we leave this to future work. Also, it would be interesting to conduct further evaluations of how parsing affects QG, as Quirk and Corston-Oliver (2006) do for machine translation, or to explore methods for adapting preprocessing tool so that they perform better for new domains (Chelba and Acero, 2004; Daumé III, 2007) or for specific tasks such as QG.

4.12 Summary

The experiments described so far intrinsically evaluated the QG system by measuring the quality of its top-ranked output, and by testing its individual components.

Results demonstrated some of the benefits of individual components compared to alternative approaches. In §4.2, we found that our method for simplified statement extraction outperformed a sentence compression approach. In §4.3, we observed that a supersense tagger performed better than a

named entity recognizer at tagging words with high-level semantic types such as PERSON that are relevant for QG. In §4.7, we found that the question ranker significantly improved the quality of top-ranked questions and performed at least as well as some alternative ranking models.

The experiments also showed that overall performance was far from perfect. In the document-level evaluation of the end-to-end system in §4.10.1, we observed that about 42% of top-ranked questions were acceptable, though this varied a great deal by the source of the text. For example, 56% of top-ranked questions from Britannica Elementary texts were rated acceptable.

In §4.10.3, we observed that transformations that rely on relatively deep linguistic analysis, such as relative clause extraction and pronoun replacement, were used somewhat more in the top-ranked questions at the sentence-level than at the document-level. Thus, it seems that for applications that require only small numbers of questions from very long texts, QG systems may want to focus on simple, high precision techniques (e.g., generating questions from the main clause, matching very specific patterns). However, for applications for which sentence- or clause-level coverage is important, QG systems may benefit from deeper analyses of sentence structure, including not only syntax (which we have focused on here) but also discourse and pragmatics.

Finally, in §4.11 we observed that the unacceptable questions produced by our system are caused by a variety of errors, suggesting a variety of avenues for future work. Though no particular error type was present in a majority of the unacceptable questions, some types (e.g., syntactic parsing errors) were more prevalent than others.

So far, we have discussed some of the challenges of QG, presented solutions to them, and intrinsically evaluated these solutions. Next, we present a user study that extrinsically evaluates our approach by exploring whether and how our QG system could help educators generate factual questions about reading passages.

Chapter 5

User Study

The experiments in the previous chapter provide evidence that techniques such as question ranking and the extraction of simplified factual statements lead to better questions than simpler approaches. The results also show that generating factual questions is still quite challenging: half or more of the top-ranked questions are ungrammatical or otherwise ill-formed. Since we want to avoid giving such bad questions to students, using the QG tool in a completely automatic fashion does not seem to be a viable option. Rather, educators need to be included in the question creation process in order to ensure a high level of quality in the output.

While the revision of system output requires time and effort, there is reason to believe that using a QG tool could be helpful by reducing the overall time and effort needed to create acceptable questions. If the quality of the system's suggestions is reasonably high, then the selection and revision of those suggestions may require less effort than completely manual question creation.

Consider the following simplified example: you are asked to write a factual question about a text on the inventor Nikola Tesla.[1] One option is to write a question on your own, which is, of course, a relatively simple task—but one that requires formulating and typing an English sentence. With a QG tool, another option would be to select a question from an imperfect list of automatically generated suggestions, such as the following (for brevity, we omit the source sentences from which they came):

(5.1) What did George Westinghouse suggest for?

[1] The example questions here were generated from the article on Tesla from the "energy KIDS" website produced by the U.S. Energy Information Administration (`http://www.eia.doe.gov/kids/`). The article was suggested by a fourth grade teacher who had used the website to find supplementary materials for a unit on energy.

(5.2) Where was Nikola Tesla born in 1856?

(5.3) When was Tesla's system used at Niagara Falls in the world's first large hydroelectric plant?

(5.4) When was Nikola Tesla born in Austria-Hungary?

We hypothesize that the availability of suggestions such as these can save time and effort in many situations. In this case, a user might judge question 5.3 to be acceptable in its current form, or might select and quickly revise question 5.2 by deleting *in 1856* to produce *Where was Nikola Tesla born?*.

Of course, if the quality of the QG system's output is too low or if the suggestions are ineffectively presented, then offering automatically generated suggestions will simply distract or frustrate users. In this chapter, we describe a QG tool and test whether teachers can use the tool to create factual questions more efficiently than when they have to write them on their own.

It is important to stress that we do not present a tool that generates all the questions that teachers need, nor one that is robust and ready for commercial deployment. Rather, we aim to demonstrate the potential for automatic QG techniques to help teachers do their work efficiently, and to study how teachers would use such a tool.

In §5.1, we describe the user interface for the QG tool, which enables efficient selection and revision of automatically generated questions. Then, in §5.2, we discuss the experimental design of a user study in which we asked teachers to write questions about a set of short articles using two variants of the QG tool's user interface—one that suggests questions generated by our QG system, and one that does not (and thus requires users to author questions on their own). §5.3 describes the mixed effects modeling approach used to analyze the results. In §5.4, we present the results, which indicate that the tool saved time and reduced mental effort but also affected the types of questions produced by participants. We also discuss the use of particular features of the interface (e.g., how often users selected system-suggested questions and then revised them).

5.1 Interface

For educators to be able to efficiently select and revise automatically generated questions, those questions must be presented, along with the source text from which they came, within a reasonable user interface. In this section, we describe a prototype QG tool we developed, with some feedback from teachers and others with teaching experience. We aimed to make the interface as simple as possible in order to avoid introducing a difficult learning curve or extraneous features that might frustrate users.

There are three main sections of the user interface screen (which is shown in Figure 5.1): an area of the screen that displays the text of a source article, an area providing lists of suggested candidate questions, and an area for the user's selected questions.[2]

The user can view lists of ranked questions at both the level of the sentence and the level of the document. To view questions at the sentence level, he or she can mouse over an individual sentence in the source article. Doing so causes the system to present a ranked list of the subset of questions that are about that sentence. To view questions at the document level, the user can click on a shortcut labeled "all questions." Doing so causes the system to present a ranked list of all of the questions about the article. Also, mousing over a suggested question causes the article text area to scroll to the source sentence for the question, allowing the user to see the context from which the question was generated.

The user can click on a question in the ranked list of suggestions to select it. A copy of the question then appears in a text box in the area for the user's selected questions. Questions there can be edited, re-ordered, or deleted. All modifications are automatically saved.

There are a few additional features worth mentioning:

- The user can perform a simple keyword search to see a ranked list of the questions in the text that contain specified words.[3]
- The user can click on shortcuts to see ranked lists of just the questions that contain one of the fol-

[2]The QG tool's front-end was implemented using javascript and the jQuery API (see `http://jquery.com/`). The tool stores questions and logs user actions in a relational database by making AJAX calls through a combination of jQuery and PHP.

[3]We create an inverted index of the words in the text to support this search. We perform case normalization but not stemming or stopword removal.

lowing question words: *who*, *what*, *where*, and *when*.

- The user can click on shortcuts to see ranked lists of just the questions from each of the following parts of the article: the first third (labeled "beginning"), the second third (labeled "middle"), and the final third (labeled "end").
- The user can add his or her own questions by clicking on a button labeled "add your own." Doing so creates a blank text area into which the user can manually type a question.
- Suggested questions whose ranking scores are in the bottom two thirds of scores for the whole article are presented in gray rather than black text to indicate that the system ranks them low (note that for sentences without any questions that are ranked highly at the document level, all will be grayed out).

Compared to the experiments in Chapter 4, there are a few changes to the way we generate and rank questions. We include yes-no questions and questions with unresolved pronouns in the question itself or in the answer phrase, but we penalize these by subtracting 1 from their predicted scores (which typically range from 1 to 5). Subtracting 1 causes these questions to appear much lower in the ranked list, usually below most of the other questions. We also subtract 1 from the score of questions whose answer's head word is a noun that is very common in the text (specifically, a noun that occurs more than 5 times and constitutes more than 5% of the noun tokens in the text).[4] Doing so downweights questions whose answer is the main subject of the article, for example.

The interface also includes the answers to each question, which are simply the answer phrase used to generate the WH questions, or "yes" for yes-no questions. When the user mouses over a question in the list of ranked suggestions, the candidate answer appears as a tooltip. Once a question is selected, the answer appears alongside the question in an editable text area.

5.2 Experimental Design

We designed an experiment to test whether teachers can use the QG tool to generate quiz questions about new texts more efficiently than if they just create the questions on their own. The experiment

[4]To account for the fact that common nouns can occur with different suffixes, we use the Porter (1980) stemming algorithm to map nouns to base forms in this step.

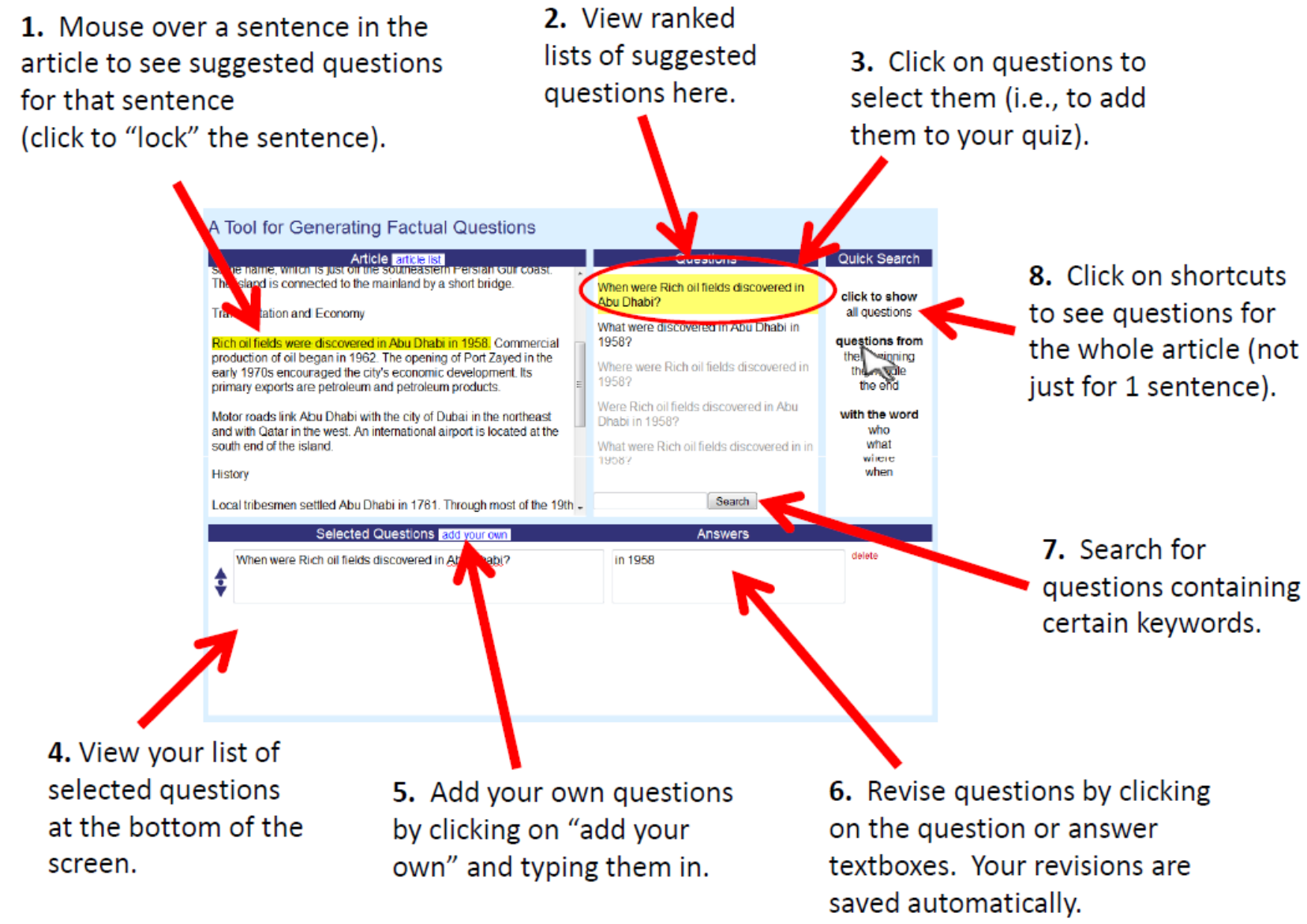

Figure 5.1: A screenshot of the question generation user interface, with the explanations of the various interface elements that were given to study participants as part of the instructions for the study.

compared two versions of the interface: the full version with automatically generated QG output as described in the previous section, and a version that was identical to the full version except that it did not present any suggested questions. When participants were using the full version, we say they were in the "QG-suggestions" condition. When they were using the version without suggestions, we say they were in the "no-suggestions" condition. In the no-suggestions condition, users had to click on the "add your own" question button and manually type questions. After discussing the participants and the texts used in the study, we describe the experimental procedure in more detail in §5.2.3.

5.2.1 Participants

A total of seventeen people with K-12 teaching experience participated in the study.[5] Participants were recruited by sending emails to schools and individual teachers with whom the author of this study had personal or professional connections. They were offered twenty-five dollars compensation for participating. Participants varied in grade level and primary subject area, though most were elementary school teachers who teach a variety of subjects, including reading, social studies, etc. One participant was a former high school math and computer science teacher, one was an elementary school reading specialist, and one was a high school history teacher. In the post-study survey, we asked participants to report their level of experience, main subject area, and grade level. Figure 5.2 shows the distribution of what participants reported as their main grade level range (i.e., the one at which they had the most experience), the distribution of subject areas that the participants reported the most experience in, and the distribution of participants' reported levels of experience.

Three participants had very briefly seen an early version of the interface, using a different text, several weeks prior to the study.

5.2.2 Texts

Participants were asked to create questions about twelve short articles from three sources:

- Four texts about international capitals from the testing set portion of the Encyclopedia Britannica

[5] One speech language pathologist who worked in an elementary school also participated in the study, but we decided not to include this person's data in our analyses since he or she was not a regular classroom teacher. We ran our analyses with this person's data to see if there would be any effect, but the results were essentially the same. E.g., with this participant included, the 99% HPD interval for the condition parameter (§5.4.1) was $(-0.55, -0.11)$ instead of $(-0.58, -0.11)$.

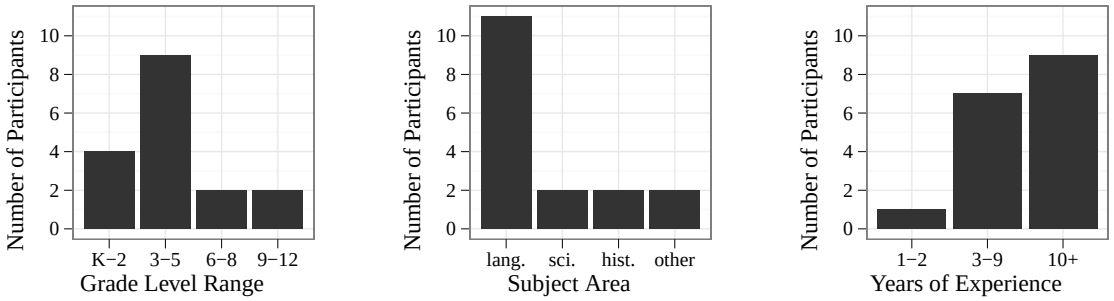

Figure 5.2: The distribution of grade levels, main subject areas, and levels of experience reported by participants. For subject areas, "lang." = reading or language arts, "sci." = science, and "hist." = history.

Elementary Edition corpus described in §4.1 of Chapter 4.

- Four texts on history and science taken from web pages that were listed as online supplementary materials for elementary school textbooks.
- Four biographical articles about famous scientists from "energy KIDS," a website aimed at elementary school students that is produced by the U.S. Energy Information Administration (see `http://www.eia.doe.gov/kids/`).

In addition, one "energy KIDS" article and one Britannica Elementary article were used as practice texts, as described in the next section.

We chose to use this set of texts rather than just texts from the corpora discussed in §4.1 of Chapter 4 because the user study mostly involved elementary school teachers. We tried to create a sample of informational texts that would be of potential educational value for K-12 teachers (Wikipedia or Britannica might also be of educational value, but probably more so for higher grade levels).

5.2.3 Procedure

The study was conducted online. Participants could use their own computers and any recent version of a major web browser (e.g., Firefox, Internet Explorer, Chrome), and could complete the study at their convenience. If necessary, participants could log out in the middle of the study and log in later

where they had stopped, but most completed the study in one sitting.[6]

The participants were first presented with a set of instructions. Participants were instructed to focus on factual questions and avoid questions that require high-level reasoning or complex inference (examples of each type were provided). Participants were also asked to check over, and if necessary revise, any computer-suggested questions they selected, and to work as quickly as they could without sacrificing the quality of the questions they produced. Participants were told that when using the full version of the interface, they could write their own questions if the system did not provide useful suggestions. They were also asked to provide correct answers to their questions. The full set of instructions is in Appendix C.

After the instructions, participants were asked to generate three questions about each of the two practice articles and then each of the twelve main articles. Thus, the user study was a within-subjects experiment in which participants created questions for six articles (plus one practice article) per condition.

The ordering of the texts, as well as the version of the interface that appeared first, was randomly chosen for each participant. Participants could not proceed to the next article until they had created three non-blank questions.

For the practice article for the full interface, additional directions asked participants to make sure that they understood and attempted to use the major features of the tool, including the ability to view suggestions for individual sentences or for the whole text, the ability to revise questions, and the ability to add their own questions.

After completing each article, participants were asked the question, "How much mental effort did you put into creating questions for the article you just saw?", adapted from research of cognitive load theory (van Gog and Paas, 2010). They responded on a five-point scale from "1 – a very low amount" to "5 – a very high amount."

After completing all of the articles, participants were asked, through drop-down menus, a series of questions about their experience as a teacher and their impressions of the system. We list the spe-

[6]To account for participants taking breaks, the time spent on an article was measured by how long a participant was logged in and working on that article, not by the difference between when a participant started and finished an article. Every ten seconds, the web page for viewing an article would send a signal to the logging server to indicate that the participant was working on that article. This robust but intermittent signal introduces a small amount of variance into time measurements. There are various other sources of variance: browser loading times, Internet connection quality, processor speeds of participants' computers, etc.

cific questions in the results section (§5.4). There was also a text box for additional comments.

Thus, for a hypothetical example participant who randomly started with the full interface, the sequence of tasks would be as follows:

1. Read instructions.
2. Write 3 questions for practice article `P2` using the full version of the QG tool interface. Respond to mental effort question.
3. Write 3 questions for practice article `P1` using the control version of the interface. Respond to mental effort question.
4. Write 3 questions for main task article `T11` using the full version of the QG tool interface. Respond to mental effort question.
5. Write 3 questions for main task article `T4` using the control version of the interface. Respond to mental effort question.
6. Repeat steps 4 and 5 for the remaining 10 of 12 main task articles, alternating between versions of the interface
7. Complete end-of-study survey questions.

In our analyses, we discard data from the practice articles, leaving us with 204 observations (twelve articles times seventeen users).

5.2.4 Outcome Measures

By analyzing log data, we measure the following outcome variables for each version of the interface:

- **Time on task**: for each article, we measure the time spent to create the three required questions.
- **Perceived mental effort**: we use the 1 to 5 reports given after each article to measure perceived effort.

- **Question acceptability and question types**: we measured the acceptability of questions produced (as in Chapter 4), as well as the type of questions created (e.g., whether or not user-created questions could have been generated by the system). We discuss these annotations in more detail in §5.4.3.

We hypothesized that when participants were in the QG-suggestions condition, they would produce similar questions compared to the no-suggestions condition, but in a shorter amount of time and with slightly less mental effort.

In addition to the outcomes measures above, we also measured how frequently certain features of the tool were used by participants as they created questions. For example, we measured how many times automatically generated questions were revised and what types of revisions were made.

5.3 Linear Mixed Effects Modeling

In this section, we give a brief introduction to the statistical modeling approach used to make inferences about the time on task and reported mental effort results. To test whether the observed differences in time and effort were due to random chance, we used linear mixed-effects models (or mixed models), an approach widely used in the social sciences, medicine, and engineering (Gilmour et al., 1995; Goldstein, 1995; Pinheiro and Bates, 2000; Baayen et al., 2008).

In this user study, there are fixed effects (i.e. discrete variables that can only take on values in the fixed set of what we observed) as well as random effects (i.e., discrete variables whose values we believe are drawn from some larger population). The fixed effects of interest are the condition the user was in when working on a particular article (QG-suggestions or no-suggestions), and how many articles the user had worked on up to and including the current article (i.e., whether the article is the first, second, third, etc.). Random effects include identifiers for each of the articles and participants, which are drawn from larger populations of articles and teachers, respectively.

In a linear mixed-effect model, in addition to the error term for individual data points, there are error terms for each level of each random effect, and these error terms are also drawn from zero-mean normal distributions. These additional error terms help us to model the different sources of variance in the observations. Following Baayen et al. (2008), we can express a mixed model as fol-

lows,

$$\begin{aligned} \mathbf{y} &= \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \boldsymbol{\epsilon} \\ \epsilon &\sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}) \\ \mathbf{b} &\sim \mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{\Sigma}) \\ \mathbf{b} &\perp \boldsymbol{\epsilon} \end{aligned} \tag{5.5}$$

where $\mathbf{y}$ is a vector of observations of a particular type (e.g., the time spent on an article), $\mathbf{X}$ is a matrix of values for fixed effects, (e.g., a 0-1 indicator for the condition), $\boldsymbol{\beta}$ is a vector of fixed-effect coefficients, $\mathbf{Z}$ is a matrix of values for random effects (e.g., 0-1 indicators for each article or user), $\mathbf{b}$ is a vector of random-effect coefficients, $\boldsymbol{\epsilon}$ is a vector of per-observation errors, $\sigma^2\mathbf{I}$ is the covariance matrix for the error terms, $\sigma^2\boldsymbol{\Sigma}$ is the variance-covariance matrix for the random effects, $\perp$ denotes independence, and $\mathcal{N}$ is multivariate normal (i.e., Gaussian) distribution. Generally speaking, $\sigma^2\boldsymbol{\Sigma}$ can be used to model covariance between random effects (e.g., covariance between a particular random slope and random intercept). In this work, we use a model with uncorrelated random effects, where $\sigma^2\boldsymbol{\Sigma}$ is a diagonal matrix. Note that $\mathbf{X}$ and $\mathbf{Z}$ are both matrices for features of the observations, but $\mathbf{X}$ includes just the fixed effects while $\mathbf{Z}$ includes just the random effects.

Random effects can be random intercepts or random slopes. An example of a random intercept is "user=john.smith", which would appear with the same value for observations involving "john.smith." An example of a random slope is "user=john.smith AND in-qg-suggestions-condition", which would be multiplied by the value of "in-qg-suggestions" for observations involving "john.smith." In this case, a random by-participant slope on the tool condition variable would allow us to model the notion that the effect of condition may vary for each participant from the average effect of condition (we included such a term in our model since such variation does appear to be present).

We used the R package `lme4` (`http://lme4.r-forge.r-project.org/`) to estimate the mixed models. The package estimates models using restricted maximum likelihood estimation (see, e.g., Gilmour et al., 1995), which corrects for known biases in the maximum likelihood estimates for variance and covariance parameters.

Following (Baayen et al., 2008), we used Bayesian Information Criterion (BIC) scores (Schwarz,

1978) and likelihood ratio tests to identify which fixed and random effects terms to include in the model. For example, we found that a random by-participant slope on condition improved BIC score, but a random by-article slope did not.

As discussed by Baayen et al. (2008), it is unknown how to compute exact p-values for mixed models with crossed random effects, which we have here in the users and articles variables. Therefore, following Baayen et al. (2008), we use the Monte Carlo Markov chain sampler in `lme4` to sample from the posterior distributions of the model parameters. After initializing the sampler with the restricted maximum likelihood estimates, we iteratively sample each of the following subsets of variables, using standard, "non-informative" priors for each: $\{\sigma^2\}$, $\{\mathbf{\Sigma}\}$, and $\{\mathbf{b}, \boldsymbol{\beta}\}$. Note that the use of non-informative, or flat, priors avoids introducing biases into the model (which are used in other types of Bayesian models).

In our analyses, we sampled for 10,000 iterations. We checked the sample posterior distributions to make sure they were close to normal. We also checked the autocorrelation plots to make sure there were no strong serial correlations in the sample chain.

From the posterior samples for each fixed effect, we compute Bayesian Highest Posterior Density (HPD) intervals (also known as credible intervals), which are the Bayesian analog to classical confidence intervals. If the 99% HPD interval for a parameter is $(0.1, 0.4)$, then the posterior probability that that parameter lies in the interval $(0.1, 0.4)$ is 0.99. Thus, if the HPD intervals for a particular parameter do not contain zero, then they support the belief that that the corresponding variable has an effect on the outcome (e.g., an effect on the time-on-task).

5.4 Results

We now turn to the results from the user study. We begin by discussing time on task and mental effort reports, and then describe question quality measurements, survey results, and interface usage statistics.

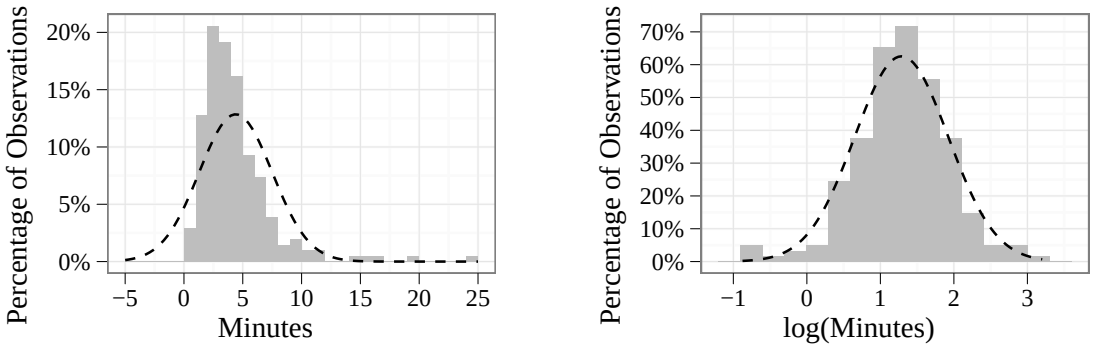

Figure 5.3: Left: a histogram showing the distribution of time on task per observation (i.e., user-article pair). Right: a histogram showing the (natural) log-transformed distribution of time on task, which is closer to normal. Dashed lines indicate expected densities of normal distributions with means and standard deviations estimated from the data.

5.4.1 Time on Task

Averaging across all observations, having QG suggestions available saved approximately one minute per article. The mean time per article in the QG-suggestions condition was 3.8 minutes, compared to 5.0 minutes in the no-suggestions condition. That is, participants spent 31% more time when in the no-suggestions condition. Equivalently, having the tool available led to a 24% time reduction compared to not having the tool available. Of course, the absolute difference in time, about 1 minute per article, is only modest.

The distribution of minutes spent per article was positively skewed, as shown in Figure 5.3. To satisfy normality assumptions, we used the natural log of the number of minutes as our response variable in our mixed model.

In addition to the general positive skew, there are a few outliers where users spent more than 15 minutes on a particular article. We tried models that excluded these outliers from the set of observations, but the models did not suggest the outliers had meaningful effects on the findings we discuss shortly, so we decided to leave the outliers in our analyses.[7]

[7]Three of the outlier observations over 15 minutes came from one participant in the no-suggestions condition. The other was from a different participant in the QG-suggestions condition. When excluding these four outliers, the 99% HPD interval for the condition parameter was $(-0.55, -0.09)$, which is almost identical to the interval for the full dataset, $(-0.58, -0.11)$.

We modeled the log-time spent on an article with fixed effects for condition and article ordering (i.e., whether the article was the first, second, etc., excluding the practice articles), a by-participant random intercept, a by-participant random slope, and a by-article random intercept.[8]

The model's estimate for the condition parameter was -0.35. Since the model was of the natural log of the time spent, we can interpret this as follows: when a user has the tool, they spend $\exp(-0.35) = 70\%$ of the amount of time they would spend without the tool. This estimated time reduction of 30% concurs with the differences we calculated before from the mean times in each condition. The 99% HPD interval for the condition parameter was $(-0.58, -0.11)$, indicating that the observed time difference between conditions was not likely due to random chance (note that this confidence interval is in log space).[9]

Though there was a reliable main effect of reduced time in the QG-suggestions condition, the tool was not equally helpful for all participants. While some participants spent substantially less time in the QG-suggestions condition, a number of participants spent about the same amount of time in both conditions, and some spent slightly less time in the no-suggestions condition, as shown in Figure 5.4. Thirteen of seventeen participants (76%) spent less time in the QG-suggestions condition, while four (24%) spent less time in the no-suggestions condition.[10]

Participants in both conditions tended to spend less time on the later articles they worked on. Including both conditions, participants spent 4.7 minutes per article for the first half of the articles, and 4.1 minutes for the second half, a difference of about thirty seconds per article. The 99% HPD interval for the article ordering number parameter (i.e., whether an article came 1st, 2nd, etc.) was $(-0.05, -0.01)$, indicating that this effect was reliable. It does not appear that participants decreased their time for later articles in the QG-suggestions condition relatively more or less than they decreased their time for later articles in the no-suggestions condition. Participants sped up by about 28

[8] The `lme4` syntax for the model was `lmer(log(minutes) ~ condition + ordering + (1|participant) + (0+condition|participant) + (1|article))`.

[9] We also tried a simpler but less sensitive significance test by averaging across the six articles for each participant, producing seventeen datapoints (one per participant) for each condition. A simple paired t-test for the differences in average time spent between conditions shows, like the mixed effects model, that the time difference is significant ($p = 0.012$). A simple paired t-test for the differences in the average mental effort (see §5.4.2) shows a marginally significant trend ($p = 0.080$).

[10] As shown in Figure 5.4, there was one user who spent 5.0 minutes less on average in the tool condition. This participant was spending much longer than most others in the no-suggestions (7.8 minutes). Excluding the data for this participant changes the 99% HPD invervals for the condition parameter only slightly to $(-0.57, -0.07)$.

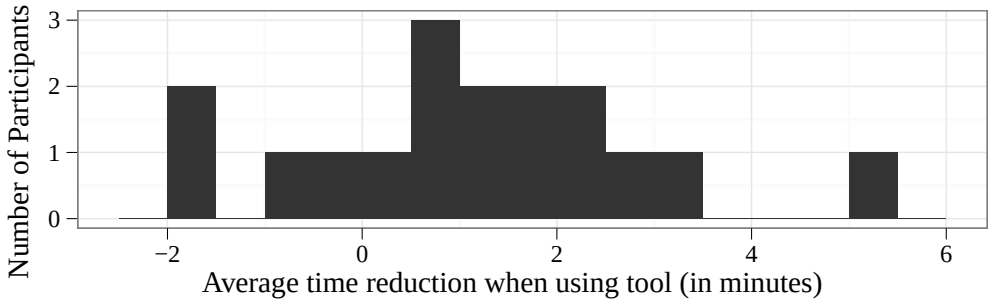

Figure 5.4: A histogram of the difference in the averages of the time spent by a participant in the two conditions. Negative values on the horizontal axis indicate that a participant spent more time in the no-suggestions condition.

seconds in the QG-suggestions condition and by about 33 seconds in the no-suggestions condition.[11]

The average time spent on an article varied a considerable amount based on the particular article. The aggregate mean across articles was 4.4 minutes, though the per-article means exhibited a standard deviation of 1.2 minutes (min. = 3.0 minutes, max. = 6.5 minutes). Average time spent varied by participant as well: the per-participant means exhibited a standard deviation of 1.9 minutes (min. = 2.2 minutes, max. = 9.7 minutes). We did not observe any clear trends that would explain why some articles or some participants took longer than others.

5.4.2 Mental Effort Reports

As with time on task, the 1 to 5 mental effort ratings made after each article tended to be lower in the QG-suggestions condition, indicating that the suggested questions not only saved time but also effort. The mean of the mental effort ratings in the QG-suggestions condition was 2.2 (std. dev. = 0.9), compared to 2.6 in the no-suggestions condition (std. dev = 1.0).

It is also worth noting that the effort ratings and time measurements were only moderately correlated, with a Kendall's rank correlation measure of $\tau = 0.43$ (Pearson's $r = 0.40$).

We fit a mixed effects model similar to the one for time on task, except we removed the article ordering number fixed effect since we did not observe any such effect when doing exploratory data

[11] We tested a model with an article ordering number-by-condition interaction term, but this alternative model had a higher (i.e., worse) BIC score, indicating that such an interaction is not supported by our data.

analysis, and removing the ordering parameter led to a more concise model with a lower (i.e., better) BIC score.[12]

The model's estimate of the condition parameter was -0.44. The 99% HPD interval for the condition parameter was $(-0.78, -0.06)$, which does not contain zero and thus indicates that the effect of condition on mental effort ratings was not likely to be due to random chance.

5.4.3 Analysis of Question Types

Now we discuss some analyses of the types of questions created by participants. We conducted these analyses to test whether there were important differences between the two conditions.

In the study, participants were instructed to create three questions about each text, and to do so as quickly as possible while maintaining quality. We ensured that participants could not continue to the next article without creating three non-blank questions, but we did not enforce that only three were created. In some cases, participants generated one or two additional questions. As such, instead of 612 total questions, 629 were created, a 2.8% increase. Roughly the same number of extra questions were created in the QG-suggestions condition (11) and the no-suggestions condition (6). Since the extra questions constituted a small percentage of the total, we simply leave them in our analyses.

We sampled and categorized a set of participant-created questions. One hundred questions were randomly sampled for each of the two conditions. For the QG-suggestions condition, the population from which the sample was drawn included all questions from that condition (i.e., questions that participants had selected from system-generated suggestions, questions that participants had selected then revised, and questions that participants had manually generated). The author of this study annotated each of the 200 questions in the sample by whether it was acceptable and by whether it could have been automatically generated, either by the QG system in its current form or with certain extensions. Using knowledge of how the QG system works, the author of this study examined the the source texts and compared participants' questions to those that the system had automatically generated for the same text. The following mutually exclusive categories were used:

- "unacceptable": The question was unacceptable according to the scheme used in the intrinsic eval-

[12] The `lme4` syntax for the model was `lmer(effort ~ condition + (1|participant) + (0+condition|participant) + (1|article))`.

uations in §4.4 of Chapter 4. Note that there was only one rater, the author of this study.

- "verbatim:" The question was generated verbatim by the QG system.
- "similar:" The question could have been generated by the system if better preprocessing (e.g., parsing or tagging) were available. Or, the system generated an acceptable question that had only a slight difference in wording.
- "transformed:" The question was a sentence-level, factual question but involved transformations not implemented in the system. Many questions in this category involved paraphrasing, lexical semantics knowledge, or relatively complex WH phrases (e.g., *what language* from *German*).
- "deep:" The question required complex inference or combining information from multiple sentences (beyond pronoun resolution), or both.

An example of a "verbatim" question written manually by a participant is question 5.6, which was also generated by the system from sentence 5.7.

(5.6) When did John Dalton die?

(5.7) When John Dalton died in 1844, he was buried with honors in England.

An example of a "similar" question is question 5.8, which was presumably created from sentence 5.9.

(5.8) Who did Marie marry?

(5.9) After four years at the Sorbonne, Marie married Pierre Curie, a well-known physicist.

The system can generate the similar but less concise question 5.10.

(5.10) After four years at the Sorbonne, who did Marie marry?

An example of a "transformed" question created by a participant is question 5.11, which was presumably created from sentence 5.12.

(5.11) Marie Curie was the first woman to win what prize?

(5.12) Marie Curie was the first woman to win a Nobel Prize in Physics.

This question uses an *in situ* question form, in which no subject-auxiliary inversion of WH-movement occurs. Such questions are not implemented in the automatic system (though an extension for *in situ* questions is certainly possible). Additionally, the question includes the question phrase *what prize*, which requires extracting the head word of the answer phrase *a Nobel Prize in Physics*. In addition to this *in situ* question type, we observed a few other factual question types not supported in the QG system (e.g., *Name three of Austria's famous composers*).

An example of a "deep" question is question 5.13, which was presumably created from sentence 5.14, but also requires knowledge about Earth's climate and the ability to make comparisons between typical wind speeds:

(5.13) Are the winds of Venus similar to those of Earth?

(5.14) Winds in the upper atmosphere blow at more than 200 mph.

As shown in Table 5.1, the distribution of question types appears to be fairly different in the two conditions.

There were unacceptable questions in both conditions, and slightly more unacceptable questions in the QG-suggestions condition. Out of the 100 questions in the no-suggestions condition, 93 were rated acceptable. Out of the 100 in the QG-suggestions condition, 87 were rated acceptable. A proportion test indicates that this difference is not statistically significant ($p = 0.239$).

Both conditions contained human-written questions that were unacceptable for various reasons, including the following question about the city of Asuncion, which is vague (since there are many port cities in South America):

(5.15) $\otimes$ What port city is located in South America?

There were also typographical errors that made some questions difficult to understand, such as in the following example.

(5.16) $\otimes$ What covers the planet covers Venus and prevents clear skies?

Question Type	no-suggestions	QG-suggestions
unacceptable	7	13
verbatim	8	48
similar	36	15
transformed	36	19
deep	13	5

Table 5.1: For the two samples of 100 questions created by participants in the no-suggestions and QG-suggestions conditions, the numbers of questions categorized by acceptability and the degree to which the current system could generate them. See the text for descriptions of the categories.

In addition, in the QG-suggestions condition, there were some unacceptable automatically generated questions that participants did not correct. For example, one participant selected (and did not revise) question 5.18, which was generated from sentence 5.18:

(5.17) $\otimes$ What is Venus Express outfitted with 128 times per second?

(5.18) Venus Express is outfitted with a camera that can take images in infrared and visual spectrums as well as a device called the Flux Gate Magnetometer, which searches for lightning signals 128 times per second.

There were more substantial differences between conditions for other categories, and the QG-suggestions condition led to simpler questions in general. Of the questions from the no-suggestions condition, 44 were either classified as "verbatim" or "similar," compared to 63 in the QG-suggestions condition. A proportion test indicates that this difference is statistically significant ($p = 0.011$). In contrast, 49 acceptable questions from the no-suggestions condition were classified as "transformed" or "deep," compared to 24 in the QG-suggestions condition (proportion test, $p = 0.005$).

Thus, it appears that suggestions from the tool led participants to select a larger proportion of simpler questions than they otherwise would have created, which raises important several issues we discuss in §5.5.

5.4.4 Exit Survey Results

In the exit survey, we asked participants the following multiple-choice questions:

- How easy to use was the version of the tool that suggested questions?

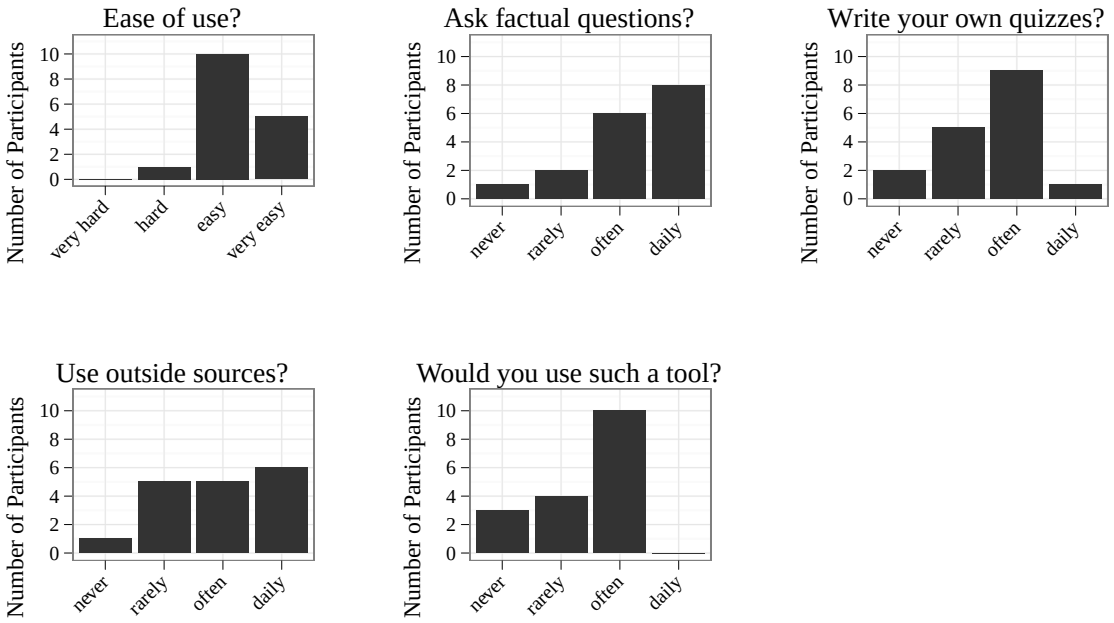

Figure 5.5: Exit survey results. The questions and anchors are abbreviated. See text for full versions.

- How often do you ask factual questions about reading assignments (e.g., to assess recognition and recall)?
- How often do you write your own quizzes about reading assignments?
- How often do you use texts from outside sources, such as the Internet or newspapers?
- How often would you use a software tool like this to create quizzes?

For the ease of use question, the options were the following: “very hard to use,” “hard to use,” “easy to use,” and “very easy to use.” For the other questions, the options were the following: “never (or almost never),” “rarely (e.g., once a month),” “often (e.g., once a week),” and “daily (or almost daily).”

The results are shown in Figure 5.5.

Almost all of the participants found the tool easy to use, indicating that the interface did not cause any major problems. Almost all also indicated that they asked factual questions very frequently,

either daily or "often." Most indicated that they write their own quizzes less frequently, about once a week for the plurality of respondents.

Use of outside texts varied a great deal. The group of seventeen participants could be split equally into three groups by whether they used outside sources once a month or less, about weekly, or about daily.

Ten of seventeen participants responded that they would probably use such a QG tool often (e.g., once a week) if it were available to them. So, there appear to be potential users for QG tools, particularly if these tools could be further refined and further developed to cover a broader range of texts and questions.

Summary of Comments

Ten of the seventeen participants chose to leave comments in the optional comments box in the exit survey. Most of the comments were very brief.

One teacher asked whether the tool could be extended to help teachers find texts as well, motivating work on educational text retrieval (Brown and Eskenazi, 2004; Miltsakaki and Troutt, 2008; Heilman et al., 2008). A few participants made small suggestions of ways to improve usability and the presentation of suggestions (e.g., not using the browser's default spell-checker).

Two explicitly mentioned difficulties in avoiding ungrammatical questions, though others surely also had issues.

One teacher said the keyword search feature was helpful, but that it would be better if one could specify also additional constraints on questions types (e.g., yes-no questions including keywords).

Two teachers commented that they do not ask many factual questions and instead focused on deeper thinking and reading skills in their classes. One of these two said that he or she uses factual questions more during oral reading practice to make sure students are successfully following along, but not much in written assessments.

5.4.5 Use of Interface Features

The various features of the tool interface in the QG-suggestion were used to varying degrees. In this section, we say that a feature was used by a participant "frequently" if he or she used that feature in

Interface Feature	Uses per Text	1+	3+	all 6
Selecting system-generated questions	2.2	16	16	10
Revising system-generated questions	0.7	15	9	0
Writing one's own questions	0.9	11	9	4
Document-level ranking	0.7	11	5	1
Question word shortcuts	0.8	11	6	2
Keyword search	< 0.1	1	0	0

Table 5.2: Usage statistics for various interface features in the QG-suggestions condition: the mean number of uses in a single article by a single participant, and the number of participants, out of 17, who used the feature in 1+ articles (excluding practice articles), 3+ articles, and in all 6 articles.

at least half (i.e., three or more) of the articles in the QG-suggestions condition.

Sixteen of seventeen participants frequently selected system-generated questions in the QG-suggestions condition. One chose not to use system-generated questions, presumably because the system did not provide enough acceptable suggestions. Additional usage statistics for this and other features are shown in Table 5.2. Table 5.2 also includes the average use of each feature, calculated as the number of times a feature was used divided by the total number of non-practice article-participant pairs in the QG-suggestions condition (102).

Nine participants frequently revised system-generated questions. Apparently they found that in many cases, an automatically generated question was close enough to being good that they could quickly and easily revise it. Also, nine participants frequently created a question of their own (i.e., without selecting or revising a suggestion from the system).

Five participants frequently used the document-level question rankings when creating questions. This count includes use of the "all questions" shortcut as well as the three shortcuts for showing top-ranked questions from the beginning, middle, and end, respectively, of a text.

Question word shortcuts, which caused top-ranked questions that included certain questions words (*who*, *what*, *when*, and *where*) to be displayed, were used frequently by six participants.

The keyword search feature, which allows users to search for questions containing certain keywords, was used only once (to search for questions containing *Edison* in the text about Nikola Tesla, Edison's rival).

There was substantial overlap in the use of some of the features. For example, the five of the par-

System Question	Teacher-Revised Version	Comments
Who is Venus commonly known as?	What is Venus commonly known as?	The system incorrectly asks a *who* question since the answer phrase is *Earth's "twin sister"*. Most of the revised questions involved simple edits such as this.
What did Marie go to in 1891?	Where did Marie go in 1891?	Here, the user corrected the slightly awkward phrasing suggested by the system. Note that the system identified the answer phrase *the Sorbonne University in Paris* as a `noun.group` instead of a `noun.location`.
Where was Thorpe born in a one-room cabin on May 28, 1888?	Where was Thorpe born on May 28, 1888?	Here, a participant simply deleted a superfluous phrase to make the system-generated question more fluent.
Who was the modern Hotel Guarani designed by?	Who designed the modern Hotel Guarani?	Here, a participant liked the content of the question, but decided to convert it from passive to active voice.
What did Paraguay declare in 1811?	When did Paraguay declare its independence?	In this case, a participant substantially changed the content of a question by altering its expected answer. Such revisions were rare.

Table 5.3: Examples of system-generated questions that were revised by user study participants.

ticipants who frequently used the document-level ranking feature also frequently used question word shortcuts. However, the use of these two features in particular does not appear to determine whether one can efficiently use the tool. There was only a weak negative correlation of $r = -0.09$ between the average amount of time a participant spent on articles in the QG-suggestions condition and the number of articles in which a participant used the document-level ranking feature, for example.

5.4.6 Revision of System-Generated Questions

One particularly interesting feature to analyze is the revision of system-generated questions. Participants frequently selected system-generated suggestions and then revised them to form better questions. Of the 317 questions created by participants about non-practice articles in the QG-suggestions condition, 70 involved selection and editing, usually by either deleting superfluous phrases or changing a few words. This represents 22% of all of the questions created in the QG-suggestions condition, and 31% of the 229 automatically generated suggestions selected by participants. Table 5.3 includes some example revisions.

5.5 Discussion

We observed that when participants were in the QG-suggestions condition, they created questions more quickly than in the no-suggestions condition, and reported less mental effort in doing so. Therefore, it appears that automatic QG tools have the potential to support teachers as they create reading practice exercises and assessments.

However, we also observed that the types of questions created in the two conditions were somewhat different. Though overall acceptability rates in the two conditions were roughly similar, participants in the QG-suggestions condition generated fewer questions involving complex paraphrasing transformations and inference—and instead selected or revised simpler questions suggested by the system. We observed this difference despite instructing participants to generate basic factual questions in both conditions, and having them go back and forth between conditions to encourage the generation of similar question types in the two conditions.

As such, we hesitate to conclude that the decreased time and effort in the QG-suggestions condition are due solely to the usefulness of the tool. The observed time and effort reductions could simply be due to the fact that participants in the QG-suggestions condition were creating simpler questions.

More generally, a potential issue is that QG tools could lead users to ask simpler questions because those are the ones that the system is most capable of generating. While such questions may be easier to create, they may or may not not be the best form of practice for students. To better understand this issue, one could study the effects of using a QG tool on student learning. One could also explore psychometric issues such as how well automatically generated questions can distinguish students at different ability levels, or how well such questions measure various specific constructs (e.g., a student's ability to decode orthography versus his or her ability to integrate or analyze information). Such issues are very important in the development of standardized tests (see, e.g., Baker, 2001; Ozuru et al., 2008; Keenan et al., 2008). We leave them, however, to future work.

We speculate that at least part of the reason that participants created somewhat more complex questions in the no-suggestions condition is a lack of awareness of the reasoning they employed while creating questions. Many questions may appear to be very literal factual questions but still

involve fairly complex reasoning. We observed a similar issue before in the example from Chapter 2 (page 40) about trying to generate the question *Who shot Lincoln?* from the Wikipedia article about Abraham Lincoln, which never explicitly mentions that John Wilkes Booth shot Lincoln. Any skilled human reader would be able to make that inference, in most cases without even realizing that an inference was necessary. It may be the case that participants created this sort of relatively complex question without realizing it.

In addition to the observed difference in the types of questions in the two conditions, there are a number of threats to the external validity of the experiment worth discussing. One issue is that the texts and questions were prepared in advance. In contrast, real users would have to expend some effort to upload a text and request questions for it (and perhaps wait a non-trivial amount of time for parsing and other processing to occur). However, the time spent preparing a text could be minimized by integrating the QG tool into a larger piece of software for managing texts and assessments, or by processing a large digital library of texts that teachers might use in advance.

Another issue is that the participants who chose to participate in the study may not represent with complete fidelity the entire population of teachers. For example, the participants may have been somewhat more computer-savvy than typical teachers.

In addition to comparing effort, time spent, and question types generated with and without automatically generated suggestions, we also observed how teachers might use a QG tool. We found that certain features were used more than others. A key finding is that many participants frequently used the features that allowed them to revise suggested questions: enabling such revisions appears to be particularly useful and important for NLP applications that generate instructional content.

The survey results indicate that participants generally found the simple interface of the tool in this study to be relatively easy to learn and use. The results also indicate that there are substantial opportunities for QG tools in real classrooms. For example, most of the participants reported frequent use of texts from outside sources such as the Internet, which typically would not include practice exercises or assessments as textbooks do.

In summary, this user study supports the claim that QG tools could save educators substantial amounts of time. Real benefits seem particularly likely to be realized if the acceptability rate of suggestions can be improved and if QG tools can be expanded to support questions that involve some-

what more complex transformations such as paraphrasing and better question phrase generation.

Chapter 6

Future Work and Conclusions

In the preceding chapters, we discussed the challenges of factual QG (Chapter 2), we described an implemented QG system that addresses some of those challenges (Chapter 3), we intrinsically tested the system (Chapter 4), and we presented a user study that explored how teachers use a QG tool based on the system (Chapter 5). In this chapter, we conclude by summarizing our results, discussing potential future work, and reiterating our contributions.

6.1 Summary of Experimental Results

In the preceding chapters, we conducted various experiments to evaluate the QG system we developed. And, where we felt it was appropriate, we conducted statistical significance tests to verify that our observations were unlikely to be due to random chance. Table 6.1 provides a concise summary of the significant results. It is important to emphasize they are not the only findings of potential interest in this work. For example, we also conducted exploratory analyses of question ranking performance (§4.8 of Chapter 4), studied the types of errors produced by the system (§4.11 of Chapter 4), and explored how the user study participants used the features of the QG tool we developed (§5.4.5 of Chapter 5).

Result	Chapter	Section
The simplified factual statement extractor provided more fluent and correct outputs and better coverage than the HedgeTrimmer sentence compression algorithm (Dorr and Zajic, 2003).	Chapter 4	§4.2
The supersense tagger (Ciaramita and Altun, 2006) used in the QG system was more accurate at predicting semantic type labels of nouns (a preliminary step for creating WH phrases) than a named entity recognizer and a most-frequent sense baseline.	Chapter 4	§4.3
Statistical ranking improved the acceptability of top-ranked questions.	Chapter 4	§4.7
The performance of the linear regression ranker that is used in the QG system was not significantly different than the performance of reasonable alternative rankers.	Chapter 4	§4.7
The acceptability rate of the top-ranked questions from the QG system, which was around 40–50% overall, varied significantly by the source of the input texts.	Chapter 4	§4.10.1
User study participants spent less time creating factual questions when they had access to suggestions from the QG system.	Chapter 5	§5.4.1
User study participants reported less mental effort when creating factual questions when they had access to suggestions from the QG system.	Chapter 5	§5.4.2
When user study participants had access to suggestions from the QG system, they selected or created more questions that involved simple transformations, and fewer questions that involved complex transformations or inference. Suggestions from the QG system did not, however, significantly affect the acceptability rate of participants' questions.	Chapter 5	§5.4.3

Table 6.1: A summary of the statistically significant results from the previous chapters and cases where there was not sufficient evidence to reject a null hypothesis. See the indicated sections for details on significance tests, experimental setups, etc.

6.2 Future Work

We have addressed only a small part of the problem of automatically generating factual questions. However, there are many areas for future work on factual QG. One particularly interesting area for future work is the exploration of alternative representations for QG. In §6.2.1, we discuss potential approaches to factual QG that rely on representations other than the syntactic phrase structure tree representation that we use in this work, and draw connections between such approaches and our work. Then, in §6.2.2, we identify some major remaining challenges that would likely be relevant regardless of the representation used by a QG system, and discuss how these challenges might be addressed.

6.2.1 Alternative Representations for Question Generation

The QG system in this work relied on phrase structure syntax trees as the main representation of the input texts. This syntactic approach is, of course, not the only possible approach to QG. Indeed, as discussed in §1.9.3 of Chapter 1, there has been some work on alternative approaches. In this section, we describe various possible representations, discuss some of their potential advantages and disadvantages, and describe how our research could inform QG approaches that use alternative representations.

Vauquois Triangle Analogy

A useful analogy can be drawn to work on machine translation (MT), another text-to-text generation problem. MT researchers have explored translation models based on various types of linguistic representations, including the following: shallow word- and phrase-based approaches that utilize large datasets to learn correspondences between token sequences (Brown et al., 1990; Koehn et al., 2003), syntactic approaches based on phrase structure or dependency trees (Wu, 1997; Yamada and Knight, 2001; Quirk et al., 2005; Graehl et al., 2008; Gimpel and Smith, 2009), and deeper approaches that involve semantic representations (Nyberg and Mitamura, 1992).

A common and useful visualization of this space of possible MT approaches is the Vauquois Triangle (see, e.g., Jurafsky and Martin, 2008, Chapter 25), an adapted version of which is shown in Figure 6.1. In case of MT, the inputs and outputs, which are represented at the lower corners of the triangle, are sentences in two different languages. For QG, the input is a text, and the output is a set of questions. The vertical dimension of the triangle represents the depth of linguistic analysis required for text analysis and generation, and the horizontal dimension shows the amount of processing needed to transfer from the input in the lower left to the output in the lower right.

Different vertical levels in the triangle represent different possible linguistic representations, with higher levels indicating greater levels of abstraction. Horizontal and vertical distances roughly represent the cost of performing operations such as parsing, transfer (or transformation), and generation. For example, a deep semantic representation would probably make parsing the input into the representation (i.e., moving far up the triangle) relatively difficult, and would probably make generating output (i.e., moving back down the triangle) also relatively difficult, but would also probably make

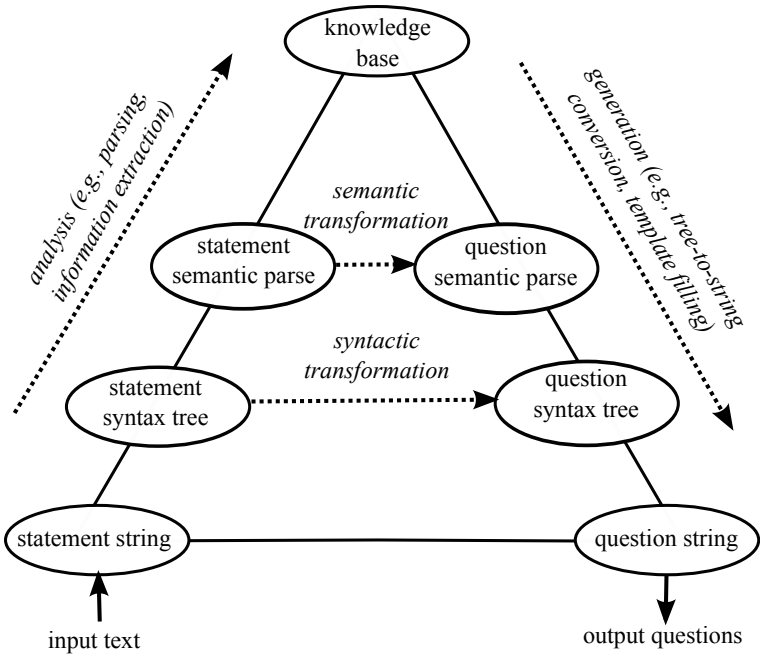

Figure 6.1: An adapted version of the Vauquois Triangle from machine translation, which illustrates the depth of linguistic analysis utilized by various text-to-text generation approaches.

transferring between inputs and outputs (i.e., moving across the top part of the triangle) relatively easy. In contrast, shallower linguistic representations would generally have low costs for parsing and generation, but high costs for transfer (e.g., because more special cases would have to be handled).

Our syntactic approach to factual QG can be characterized as follows: a moderately difficult analysis step using well-studied but imperfect tools for syntactic parsing, a rule-based transfer step leveraging linguistic knowledge about syntactic embedding and question formation, and a simple generation step (i.e, reading off the leaves of phrase structure tree in left-to-right order). Other approaches to QG have explored similar syntactic representations (Kunichika et al., 2004; Gates, 2008) (though without a statistical ranking component).

Another possible approach to QG would be to use a shallow parser (Sha and Pereira, 2003) or similar tool to extract a shallower syntactic representation, as discussed by Mitkov and Ha (2003). Such an approach would have a simpler parsing step than a full syntactic parsing approach, and a similarly straightforward generation step. However, the transfer step (i.e., converting complex declarative sentences into questions) would be more difficult in general. Using shallow parses to ex-

tract questions from the variety of linguistic constructions that one might encounter in realistic texts would likely require handling a large number of special cases. Also, incorporating linguistic knowledge (e.g., about WH-movement) seems more difficult in such an approach.

One might also consider QG from somewhat deeper semantic representations such as the output of a semantic frame parser (Das et al., 2010), or a semantic role labeler (Gildea and Jurafsky, 2002) as in recent work on QG by Mannem et al. (2010). With a semantic representation, analysis of input sentences would be more challenging because semantic parsers are typically more error-prone than syntactic parsers. Roughly speaking, while the performance of syntactic parsers is usually around 90% (Klein and Manning, 2003), the performance of semantic parsers and semantic role labelers is usually much lower, around 60–70% (Das et al., 2010) (of course, the respective evaluation metrics are not directly comparable and performance varies by system, formalism, domain, etc). Transforming semantic representations of the input into question representations might be simple if these representations fit well with the needs of QG, but semantic representations might be overly abstract, requiring a variety of templates for generating questions. For example, one might have to devise a set of question templates for the 900+ different frames in FrameNet (Baker et al., 1998). In contrast, the syntactic trees we employ provide a flexible, relatively low-level representation of natural language.[1]

Information Extraction for Question Generation

A particularly intriguing approach to QG would be to adapt methods for extracting structured information (e.g., instances of related predicates) to populate a knowledge base, as in work by Carlson et al. (2010), for example. We refer to this as an information extraction (IE) approach—though, of course, other paradigms such as semantic role labeling and named entity recognition can also be viewed as information extraction. In such an approach, one would extract a knowledge base from a particular text and then generate questions from that knowledge base, which can be viewed as going slightly further up the Vauquois Triangle than semantic role labeling or syntactic parsing.

An IE approach might facilitate the generation of deeper questions that abstract away from particular lexical items and involve some basic inference. For example, from the phrase 6.1, one might

[1] Alternative representations could also be used to define features for question ranking. For example, one could use a primarily syntactic approach to generate questions, as we do here, but then include ranking features for the presence of various semantic frames.

extract predicate instances such as example 6.2, and then generate an output such as question 6.3.

(6.1) . . . Chicago Bulls shooting guard Michael Jordan . . .

(6.2) **player-plays-for-team**(*Michael Jordan*, *Chicago Bulls*)

(6.3) What team does Michael Jordan play for?

In addition, with such abstract predicates, one might be able to apply general inference procedures to support even deeper questions.

However, there are a number of obstacles to using an IE approach for QG. First, coverage of predicate types is an issue. Many IE systems are fairly limited in the domains and types of predicates they cover. For example, the knowledge base for the NELL system (Carlson et al., 2010) includes only 357 predicate types, covering domains such as sports, business, and geography.[2]

Using the training set from our QG experiments (§4.5 of Chapter 4), we ran a simple experiment to test the quantity of categories and relations (i.e., unary and binary predicate instances) that one might extract using the set of extraction patterns from the NELL system. Each NELL pattern takes one of the following four forms: *ARG1 unary-pattern*, *unary-pattern ARG1*, *ARG1 binary-pattern ARG2*, or *ARG2 binary-pattern ARG1*. In these patterns, *unary-pattern* and *binary-pattern* are text spans such as *locations such as* or *shooting guard* that correspond to a predicate type (e.g., *location*, *athlete-plays-for-team*) and match spans of text between possible arguments to that predicate. The NELL system identifies noun phrases that can serve as arguments and places various restrictions on the lengths and types of these arguments. For our experiment, in order to estimate an upper bound for the number of relations extracted, we simply identified matches of the *unary-pattern* and *binary-pattern* text spans without restricting the text spans that serve as arguments.

From running the 85,325 NELL extraction patterns over our training set of 40 texts (consisting of about 75,000 words), 869 predicate instances were extracted—an average of 22 instances per text, which is smaller than the average number of sentences per text (60). Of these 869 instances, 812 were binary relations and 57 were categories (i.e., unary relations). We observed that the great majority of these instances were invalid. For example, assuming that good noun phrase identification

[2]We downloaded the NELL extraction patterns for iteration 185 from `http://rtw.ml.cmu.edu/rtw/resources` in early January, 2011.

could identify the spans of arguments, predicate instance 6.4 would be extracted from phrase 6.5 because of the pattern 6.6 for the **athlete-plays-for** binary relation.

(6.4) **athlete-plays-for-team(***completion*, *Spire***)**

(6.5) . . . until the completion of the Chicago Spire . . .

(6.6) *ARG1* of the Chicago *ARG2*

We ran a second test in which we restricted the set of extraction patterns to only those labeled with 0.5 or greater probability in the knowledge base. Using only these 54,479 high-confidence patterns, a total of 60 predicate instances were extracted from the 40 texts, with only 8 of them being binary relations. An example of one such binary relation is relation 6.7, which was extracted from sentence 6.8 using pattern 6.9.

(6.7) **location-located-within-location(***Nicosia*, *Cyprus***)**

(6.8) The divided city of Nicosia is the capital of Cyprus.

(6.9) *ARG1* is the capital of *ARG2*

A general point worth mentioning is that systems such as NELL address the problem of macro-reading, or extracting information from large quantities of text by relying on redundancy and agreement between disparate sources of evidence (Mitchell et al., 2009). In contrast, factual QG, at least as we have framed it, deals with the problem of micro-reading, or extracting information from a single text by deeply analyzing linguistic structure. The experiment above is limited in that it uses patterns developed for macro-reading in a very simple way to address a micro-reading task, but at least it illustrates that coverage would be an important issue in an IE approach to QG.

There are IE techniques for micro-reading, of course, but they have typically been limited to a very small set of predicates (e.g., named entity recognition systems identify instances of a few general categories such as persons, locations, and organizations). Recent work on "distant supervision" has attempted to expand the set of predicates covered by IE systems in various ways (Mintz et al., 2009; Riedel et al., 2010; Hoffmann et al., 2010), such as by leveraging data from Wikipedia (`http://www.wikipedia.org/`) and Freebase (`http://www.freebase.com/`). Such

approaches might be able to support a larger variety of predicates, but they typically focus on macro-reading.

In addition to coverage, a second issue is that most IE systems such as NELL focus on unary and binary predicates, limiting the variety of questions one might generate. *Where* and *when* questions in particular often involve more than just two arguments (e.g., *When was Thomas Jefferson elected President?*, *Where did Robert E. Lee surrender to Grant?*). Also, with only one or two arguments per predicate, the issue of vagueness when taking information out of context (§2.3.1 of Chapter 2) may be even more prevalent than in a syntactic approach. There has been some research on event and case frame extraction techniques that address more complex predicates (e.g., Lehnert et al., 1992; Riloff and Schmelzenbach, 1998; Aone and Ramos-Santacruz, 2000), but most of that work has been fairly domain-specific (e.g., extracting information about terrorist attacks).

A third issue is that even if one could extract complex predicate instances for a variety of domains, creating questions from these predicates is nontrivial. For example, assuming an IE system was able to extract a relation such as **elected(***Thomas Jefferson*, *president*, *1800***)**, one would still have to convert that abstract representation into a question. To do so, one might have to specify or learn question templates for each relation type. Thus, while IE technologies appear to have considerable potential for supporting deeper QG, there are a number of important challenges yet to be solved.

In this section, we have discussed various alternatives to our syntactic approach to QG, and their various advantages and disadvantages. Developing systems that use these representations is an interesting challenge, but a rather complex one involving considerable engineering effort, and so we leave it to future work.

However, we observe that QG systems using linguistic representations other than phrase structure trees could still benefit from some of the findings we have presented. For example, the overgenerate-and-rank QG framework could be adapted to work with other syntactic or semantic representations. In particular, the ranking step and most of its features could be re-used or adapted.

6.2.2 Remaining Challenges

It should be clear from the experimental results and error analysis in Chapter 4 that many challenges in automatic factual QG remain to be solved. Though current QG technologies show promise with

respect to enabling useful educational applications, more research is needed, both to improve the acceptability of automatically generated questions and to extend the range of question types that can be reliably produced.

In Chapter 2, we provided a broad categorization of the challenges in QG. Of course, all of these challenges are potential areas for future work, particularly those such as the effects of implicit discourse relations (§2.3.2) that we did not focus on in our research. Some of these challenges, such as parsing of syntactic structures (§2.2.1), are general problems in computational linguistics that affect a variety of NLP applications. In contrast, some challenges, such as the generation of WH phrases (§2.1.1) are relatively specific to QG—though research on them might inform work in related fields (e.g., question answering, text-to-text generation).

In this section, we identify a few particularly interesting challenges specific to QG and discuss what sort of future work might be needed to address them. We identify challenges for each of the major areas discussed in Chapter 2 (i.e., lexical challenges, syntactic challenges, discourse challenges, and challenges related to the use of QG tools).

Mapping Answers to Question Phrases

One of the most prevalent lexical challenges we encountered in our experiments was the difficulty of generating appropriate WH phrases for a given answer phrase (§2.1.1 of Chapter 2 and §4.11 in Chapter 4). For example, while the techniques we discussed could map an answer phrase such as *Thomas Jefferson* to the question phrase *who*, these techniques would often produce vague or awkward question phrases depending on the context. In many cases, more specific question phrases would be needed to make questions fluent and clear. For example, consider the following two questions.

(6.10) Who was born in Shadwell, Virginia?

(6.11) Which U.S. president was born in Shadwell, Virginia?

Question 6.10 might be acceptable in the context of an article about the lives of U.S. presidents but would almost certainly be vague in the context of an article about the history of the city of Shad-

well, Virginia. In contrast, question 6.11 would likely be acceptable in both of those contexts (but might be too specific in others).

There are at least two major parts of this problem. First, one must map potential answer phrases to potentially relevant question phrases. This process might require considerable lexical semantics knowledge—for example, that Jefferson was president, that presidents are leaders, that leaders are people, etc. Sources such as WordNet (Miller et al., 1990), Wikipedia (`http://www.wikipedia.org/`), and Freebase (`http://www.freebase.com/`) may be able to provide such information, but applying them to QG seems nontrivial.

Second, one must decide which of many potential question phrases are most appropriate for a particular context. To address this issue, a system would need to identify whether a question phrase is too vague, permitting answers other than the intended one (e.g., the names of people other than Thomas Jefferson who were born in Shadwell, Virginia), or too specific, incorporating information not present in the context (e.g., *which founding father* might not be an appropriate way to refer to Jefferson if the text did not label Jefferson as a founding father). Question answering technologies might help to identify such alternative answers. Or, an information theoretic approach (e.g., measuring the entropy over potential answers) might be useful.

Finally, two additional issues related to question phrase creation are the following: that one should ensure the fluency of the resulting questions, and that one should avoid questions that might provide inappropriate clues, such as in question 6.12.

(6.12) Which 3rd U.S. President was born in Shadwell, Virginia?

Learning to Extract Factual Statements

As described in §2.2.2 of Chapter 2, another important problem is how to extract factual statements from the various syntactic constructions that appear in complex sentences. While we showed that our technique for simplified factual statement extraction produces fluent, correct, and concise statements from complex sentences, future work could improve upon the robustness and coverage of our approach, and also perhaps decrease the amount of linguistic knowledge needed to develop the system.

One avenue for improving robustness and flexibility would be to incorporate some of the relevant work on statistical models for related text-to-text generation problems (Clarke, 2008; Gimpel and

Smith, 2009; Zhu et al., 2010; Woodsend et al., 2010). For example, Clarke (2008) uses using integer linear programming to encode constraints based on linguistic knowledge into a machine learning approach for sentence compression. Also, rich formalisms such as quasi-synchronous grammar (Smith and Eisner, 2006; Gimpel and Smith, 2009; Woodsend et al., 2010) could facilitate the learning of complex transformations from data while still allowing the incorporation of linguistic knowledge about questions. Such techniques might enable the learning of more robust mappings from complex sentences to simpler statements. Of course, an important issue is that such techniques often require supervision in the form of labeled data, and such data may be difficult to acquire.

Statistical approaches might also improve the fluency and coverage of simplified factual statement extraction. In our experiments, we found that our system did not always extract statements that led to the most concise and natural questions possible. For example, an extracted statement such as example 6.13 would lead to a reasonable but not particularly natural-sounding question such as 6.14.

(6.13) Thomas Jefferson was born in Shadwell, Virginia in 1743.

(6.14) Where was Thomas Jefferson born in 1743?

It would be advantageous in such cases to perform some additional simplification to remove unnecessary modifiers in order to produce outputs such as example 6.15 and question 6.16.

(6.15) Thomas Jefferson was born in Shadwell, Virginia.

(6.16) Where was Thomas Jefferson born?

Statistical modeling might also be used to learn statement-to-question transformations, but the issue of where to find relevant labeled or aligned data seems even more challenging than with simplified statement extraction. Work by Higgins (2003) and others on identifying long-range dependencies (i.e., gaps) in sentences with WH complements may be relevant.

Resolving Vagueness out of Context

A major challenge in discourse processing for QG is identifying and resolving vagueness that occurs when taking information out of context. As discussed in §2.3.1 of Chapter 2, the sentences in a text

will often assume that the reader can recall the context described in the preceding discourse. If QG systems ignore this phenomenon and treat sentences as stand-alone units, they will generate unacceptable, vague questions (e.g., $\otimes$ *Who surrendered?*)

Research on more general discourse processing techniques, such as work on event coreference (Chen et al., 2010), could enable QG systems to generate questions that incorporate temporal or locational information from previous sentences (e.g., *Who surrendered at Appomattox Court House in 1865?*). However, even systems for pronoun resolution, which is a comparatively much simpler task than event coreference, produce a considerable quantity of errors, so it is unclear whether such technologies would be sufficiently accurate to be useful.

Improving Usability of Question Generation Tools

The usability of QG tools is another important open challenge (§2.4.2 of Chapter 2). QG tools may be able to help educators, but only if those tools are highly usable and have features that teachers find relevant.

One potential opportunity in this area is to develop a system for sharing the questions created by users so that future users working with the same text could re-use those questions. Such manually created and checked questions could be presented first in the tool's ranked list, or via some special interface element. In such a collaborative environment, a QG system could provide "seed" questions for new texts. Also, teachers could share questions that go beyond factual knowledge to address reading strategies or higher-level cognitive processes. Another intriguing possibility is using data collected from teachers using a QG tool to improve an existing QG system—for example, by treating questions that teachers selected as positive examples when training a question ranker.

Another potential opportunity is the integration of QG tools into a larger system for managing the texts and assessments used in a particular curriculum. Factual QG is only one technology with potential utility for educators. Others include educational text retrieval (Brown and Eskenazi, 2004; Heilman et al., 2008; Miltsakaki and Troutt, 2008), automated grading (Nielsen et al., 2008; Sukkarieh and Bolge, 2008; Bailey and Meurers, 2008; Mohler and Mihalcea, 2009), and automatic QG for other question types such as vocabulary questions (Brown et al., 2005) and cloze-format reading comprehension questions (Mostow and Chen, 2009). Integrating several of these technolo-

gies could provide substantial benefits in terms of usability and efficiency. For example, a teacher might use a text retrieval tool to search a digital library for texts at a particular reading level, then use a factual QG tool along with other QG tools to generate a reading quiz, and then finally grade student responses to that quiz using an automated grading tool.

It is worth noting that many of the ideas described in this section were suggested by teachers who participated in the user study described in Chapter 5.

Deeper Questions and Other Genres

One intriguing area for future work is the generation of deeper questions by performing *semantic* transformations, rather than just syntactic ones. A potential way to support deeper questions in our framework is to incorporate semantic transformations in stage 1, in place of or in addition to the simplified factual statement extraction step. For example, one could incorporate paraphrase generation techniques (Callison-Burch, 2007) to create questions with less lexical overlap with the input text, as in the following example.

- **Input Sentence:** John Wilkes Booth, who assassinated Abraham Lincoln, was a famous actor in addition to being a Confederate sympathizer.
- **Simplified Factual Statement (stage 1, §3.2 of Chapter 3):** John Wilkes Booth assassinated Abraham Lincoln.
- **Paraphrase Generation (stage 1, potential future work):** John Wilkes Booth killed Abraham Lincoln.
- **Question Creation (stage 2, §3.3 of Chapter 3):** Who killed Abraham Lincoln?

New ranking features could also be incorporated to model how such transformations affect question quality.

A problem closely related to paraphrase modeling is textual entailment. Though researchers have mainly focused on recognizing textual entailment (Dagan et al., 2005), future research may also enable the *generation* of various kinds of entailed sentences. If so, such transformations could also be supported in stage 1 of our framework—for example, as follows.

- **Input Sentence:** Booth jumped into the box and aimed a single-shot, round-slug .44 caliber Henry Deringer at Lincoln's head, firing at point-blank range.
- **Textual Entailment Generation (stage 1, potential future work):** Booth killed Abraham Lincoln.
- **Question Creation (stage 2, §3.3 of Chapter 3):** Who killed Abraham Lincoln?

In addition to deeper questions, future work could also explore QG from texts in other genres. In this work, we focused on informational texts such as encyclopedia articles, but factual QG from narratives or other types of texts would also be interesting and useful. Challenges related to phenomena such as metaphorical language and complex discourse structures could be more prevalent in other genres.

Automated Evaluation Metrics

Another area for future work in QG is the development of efficient evaluation techniques. It would be very useful to develop automatic performance measures for QG similar to the BLEU (Papineni et al., 2002) metric that is commonly used to evaluate machine translations, or the ROUGE (Lin, 2004) metric commonly used to evaluate summaries. These metrics compare system outputs to reference (i.e., gold standard) outputs with simple, easily computed statistics such as n-gram overlap.

Such automated metrics, however, require significant initial human effort for creating sets of reference outputs. For machine translation, such reference outputs are often readily available since human translators already translate many of the texts used in translation research (e.g., parliamentary proceedings). Clever evaluation approaches may also provide these reference translations in some cases: for example, as mentioned in §4.2.4 of Chapter 4, Zhu et al. (2010) employ BLEU to test their sentence simplification system by using noisily aligned sentences from Simple English Wikipedia as reference simplified versions of sentences from Wikipedia. However, for other simplifications experiments or for QG experiments, acquiring reference outputs would likely be difficult. Textbooks or standardized tests are a potential source of gold standard questions, but questions from such sources might not match up well with the types of questions that a system generates (e.g., a textbook might contain many questions involving inference or reading strategies).

In addition, adapting a metric such as BLEU for QG evaluations is non-trivial since there are multiple—and often quite different—questions that one might ask for a given sentence. While for machine translation, the good translations for a sentence can be considered paraphrases of each other, for QG, the acceptable questions for a sentence can be very different (e.g., *Who did John meet?* versus *Who met Susan?*), particularly for sentences that convey multiple facts via relative clauses, conjunctions, etc.

Languages Other than English

Finally, it is worth mentioning that this work has focused only on factual QG from texts written in the English language. Extending research on QG to other languages is certainly feasible, particularly to languages for which accurate parsers and other NLP tools are available.

New languages would likely present new challenges for QG. Let us consider Japanese as an example. In Japanese, question phrases are commonly *in situ* rather than being at the beginning of the main clause as in English (e.g., *Thomas Jefferson was born where?* rather than *Where was Thomas Jefferson born?*). Certain types of questions are still ungrammatical due to syntactic constraints (Takahashi, 1993), but these likely have somewhat different realizations than in English. Japanese is also a pro-drop language (particularly in spoken dialects), meaning that anaphors can be omitted in certain circumstances in which they can be inferred by listener. For example, $*$*Was born in Shadwell, Virginia* might be a valid English sentence if English were a pro-drop language. This characteristic of Japanese might raise new challenges related to vagueness and discourse processing.

There are, of course, many other such issues in Japanese and in other languages.

Another example is the topic-focus distinction (Sgall et al., 1973; Hajičová et al., 1995; Gundel and Fretheim, 2006). The topic of a sentence is what is being discussed, and the focus (also called the comment) is typically new information about the topic. English syntax does not prominently encode topic-focus structure, though some constructions, such as cleft constructions and passivization, can be used to emphasize what the focus is (e.g., *John* is the focus in the cleft sentence *It was John who I spoke with*). In some languages, however, such as Czech and Russian, topic-focus is more systematically realized. For languages such as these, and perhaps for others such as English as well, QG techniques might benefit from including question ranking features such as whether the answer to a

question was the topic or focus of the sentence from which the question was generated (the intuition in this case being that questions about foci, which express new information, might be more important).

6.3 Summary of Contributions

In this dissertation, we have made the following contributions to the literature on natural language processing and educational technologies:

We analyzed various linguistic and computational challenges involved in automatic factual QG, a relatively new problem that is related to a number of other text-to-text generation problems. Our analysis may provide future QG researchers, as well as those working on related problems, with a useful roadmap.

We described an overgenerate-and-rank approach to QG that enables us to use existing NLP tools for syntactic analysis, to leverage linguistic knowledge about question formation, and to apply machine learning to rank questions. As an initial step in this framework, we described a component for extracting simplified factual statements, which can then separately be converted into questions, from complex input sentences.

A key feature of our approach is statistical question ranking. We use a statistical model of question quality to rank candidate questions so that the best questions are more likely to be shown to users first. In experiments, we found that ranking roughly doubles, on average, the acceptability of the top-ranked questions for a given text.

We also presented the results of a user study involving teachers that demonstrated the potential for QG technologies, and NLP technologies more generally, to have an impact on education. We found that a QG tool enabled teachers to create factual questions more quickly and with less reported mental effort than when creating questions on their own. However, the tool also affected the types of questions created by users, leading to somewhat simpler questions. In addition, we studied how teachers might use such tools. Specifically, we analyzed how participants made revisions to automatically generated questions and measured how often certain features of the interface were used.

As a more general point, this work supports the idea that natural language processing techniques

have the potential to have a positive impact on educational technologies, supporting educators in the efficient use of new digital text resources, and thereby facilitating the delivery of instruction that matches students' interests and learning needs.

Bibliography

Ainsworth, S. (2004). Evaluating the REDEEM authoring tool: Can teachers create effective learning environments? *International Journal of Artificial Intelligence in Education*, 14(3).

Aleahmad, T., Aleven, V., and Kraut, R. (2008). Open community authoring of targeted worked example problems. In *Proc. of Intelligent Tutoring Systems*.

Anderson, J., Corbett, A., Koedinger, K., and Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2).

Aone, C. and Ramos-Santacruz, M. (2000). REES: A large-scale relation and event extraction system. In *Proc. of the Conference on Applied Natural Language Processing*.

Baayen, R. H., Davidson, D. J., and Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59.

Bailey, S. and Meurers, D. (2008). Diagnosing meaning errors in short answers to reading comprehension questions. In *Proc. of the Third Workshop on Innovative Use of NLP for Building Educational Applications*.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proc. of COLING*.

Baker, F. B. (2001). *The Basics of Item Response Theory*. ERIC Clearinghouse on Assessment and Evaluation, 2nd edition.

Bar-Haim, R., Dagan, I., Greental, I., Szpektor, I., and Friedman, M. (2007). Semantic inference at

the lexical-syntactic level for textual entailment recognition. In *Proc. of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Barzilay, R. and Elhadad, N. (2003). Sentence alignment for monolingual comparable corpora. In *Proc. of EMNLP*.

Barzilay, R. and Lapata, M. (2008). Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1).

Beigman Klebanov, B., Knight, K., and Marcu, D. (2004). Text simplification for information seeking applications. *On the Move to Meaningful Internet Systems*.

Bloom, B. (1956). *Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive Domain*. Longmans, New York, NY.

Bormuth, J. R. (1967). *Cloze readability procedure*. University of California, Los Angeles. CSEIP Occasional Report No. 1.

Boyer, K. E., Ha, E. Y., Wallis, M. D., Phillips, R., Vouk, M. A., and Lester, J. C. (2009). Discovering tutorial dialogue strategies with hidden Markov models. In *Proc. of AIED*.

Brown, J. and Eskenazi, M. (2004). Retrieval of authentic documents for reader-specific lexical practice. In *Proc. of InSTIL/ICALL Symposium*.

Brown, J., Frishkoff, G., and Eskenazi, M. (2005). Automatic question generation for vocabulary assessment. In *Proc. of HLT/EMNLP*.

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2).

Burges, C., Ragno, R., and Le, Q. V. L. (2006). Learning to rank with nonsmooth cost functions. In *NIPS*.

Callison-Burch, C. (2007). *Paraphrasing and Translation*. PhD thesis, University of Edinburgh.

Carbonell, J. and Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. of SIGIR*.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E. R. H., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Proc. of AAAI*.

Carlson, A., Gaffney, S., and Vasile, F. (2009). Learning a named entity tagger from gazetteers with the partial perceptron. In *Proc. of the AAAI Spring Symposium on Learning by Reading and Learning to Read*.

Carnie, A. (2006). *Syntax: A Generative Introduction*. Wiley-Blackwell, 2nd edition.

Carreras, X. and Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proc. of CoNLL*.

Chakrabarti, S., Khanna, R., Sawant, U., and Bhattacharyya, C. (2008). Structured learning for non-smooth ranking losses. In *Proc. of KDD*.

Chang, M., Goldwasser, D., Roth, D., and Srikumar, V. (2010). Discriminative learning over constrained latent representations. In *Proc. of NAACL*.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proc. of NAACL*.

Chelba, C. and Acero, A. (2004). Adaptation of maximum entropy classifier: Little data can help a lot. In *Proc. of EMNLP*.

Chen, B., Su, J., and Tan, C. L. (2010). Resolving event noun phrases to their verbal mentions. In *Proc. of EMNLP*.

Chiang, D., Marton, Y., and Resnik, P. (2008). Online large-margin training of syntactic and structural translation features. In *Proc. of EMNLP*.

Chomsky, N. (1957). *Syntactic Structures*. Mouton.

Chomsky, N. (1973). Conditions on transformations. In Anderson, S. R. and Kiparsky, P., editors, *A Festschrift for Morris Halle*. Holt, Rinehart and Winston.

Chomsky, N. (1977). On Wh-Movement. In Culicover, P. W., Wasow, T., and Akmajian, A., editors, *Formal Syntax*. Academic Press.

Chomsky, N. (1981). *Lectures on Government and Binding*. Foris Publications.

Chomsky, N. and Lasnik, H. (1977). Filters and 'control'. *Linguistic Inquiry*, 8.

Ciaramita, M. and Altun, Y. (2006). Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proc. of EMNLP*.

Clarke, J. (2008). *Global Inference for Sentence Compression: An Integer Linear Programming Approach*. PhD thesis, University of Edinburgh.

Collins, M. (1999). *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania.

Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proc. of ICML*.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

Copestake, A., Flickinger, D., Pollard, C., and Sag, I. A. (2005). Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(4).

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *JMLR*.

Crammer, K. and Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *JMLR*.

Dagan, I., Glickman, O., and Magnini, B. (2005). The PASCAL recognising textual entailment challenge. In *Proc. of the PASCAL Challenges Workshop on Recognising Textual Entailment*.

Das, D., Schneider, N., Chen, D., and Smith, N. A. (2010). Probabilistic frame-semantic parsing. In *Proc. of NAACL-HLT*.

Das, D. and Smith, N. A. (2009). Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*.

Daumé III, H. (2007). Frustratingly easy domain adaptation. In *Proc. of ACL*.

del Soldato, T. and du Boulay, B. (1995). Implementation of motivational tactics in tutoring systems. *International Journal of Artificial Intelligence in Education*, 6(4).

Dolan, B., Quirk, C., and Brockett, C. (2004). Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proc. of COLING*.

Dolan, W. B. and Brockett, C. (2005). Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

Dorr, B. and Zajic, D. (2003). Hedge Trimmer: A parse-and-trim approach to headline generation. In *Proc. of the HLT-NAACL Workshop on Automatic Summarization*.

Echihabi, A. and Marcu, D. (2003). A noisy-channel approach to question answering. In *Proc. of ACL*.

Erkan, G. and Radev, D. R. (2004). Lexrank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1).

Fazly, A., Cook, P., and Stevenson, S. (2009). Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics*, 35(1).

Feng, M. and Heffernan, N. T. (2006). Informing teachers live about student learning: Reporting in the Assistment system. *Technology, Instruction, Cognition, and Learning*, 3.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proc. of ACL*.

Foster, J. (2010). cba to check the spelling. In *Proc. of NAACL-HLT*.

Freund, Y. and Schapire, R. E. (1999). Large margin classification using the perceptron algorithm. *Machine Learning*.

G. Angeli, P. L. and Klein, D. (2010). A simple domain-independent probabilistic approach to generation. In *Proc. of EMNLP*.

Gall, M. (1984). Synthesis of research on teachers' questioning. *Educational Leadership*, 42(3).

Gao, J., Qi, H., Xia, X., and Nie, J. (2005). Linear discriminant model for information retrieval. In *Proc. of SIGIR*.

Gates, D. M. (2008). Generating reading comprehension look-back strategy questions from expository texts. Master's thesis, Carnegie Mellon University.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B., editors (2007). *The third PASCAL recognizing textual entailment challenge*.

Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28.

Gilmour, A. R., Thompson, R., and Cullis, B. R. (1995). Average information REML: An efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics*, 51.

Gimpel, K. and Smith, N. A. (2009). Feature-rich translation by quasi-synchronous lattice parsing. In *Proc. of EMNLP*.

Glickman, O., Dagan, I., and Koppel, M. (2005). A probabilistic classification approach for lexical textual entailment. In *Proc. of AAAI*.

Goldberg, A. (2006). *Constructions at Work: The Nature of Generalization in Language*. Oxford University Press, New York.

Goldstein, H. (1995). *Multilevel statistical models*. Arnold.

Graehl, J., Knight, K., and May, J. (2008). Training tree transducers. *Computational Linguistics*, 34.

Graesser, A., Rus, V., and Cai, Z. (2008). Question classification schemes. In *Proc. of the Workshop on Question Generation*.

Graesser, A. C. and Black, J. B. (1985). *The psychology of questions*. Erlbaum.

Graesser, A. C., Chipman, P., Haynes, B. C., and Olney, A. (2005). Autotutor: an intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, 48(4):612–618.

Graesser, A. C. and Person, N. K. (1994). Question asking during tutoring. *American Educational Research Journal*, 31.

Gundel, J. K. and Fretheim, T. (2006). Topic and focus. *The Handbook of Pragmatics*.

Haghighi, A. and Klein, D. (2009). Simple coreference resolution with rich syntactic and semantic features. In *Proc. of EMNLP*.

Hajičová, E., Skoumalová, H., and Sgall, P. (1995). An automatic procedure for topic-focus identification. *Computational Linguistics*, 21(1).

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1).

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

Heilman, M. and Eskenazi, M. (2007). Application of automatic thesaurus extraction for computer generation of vocabulary questions. In *Proc. of the SLaTE Workshop on Speech and Language Technology in Education*.

Heilman, M. and Smith, N. A. (2009). Question generation via overgenerating transformations and ranking. Technical Report CMU-LTI-09-013, Language Technologies Institute, Carnegie Mellon University.

Heilman, M. and Smith, N. A. (2010a). Extracting simplified statements for factual question generation. In *Proc. of the Third Workshop on Question Generation*.

Heilman, M. and Smith, N. A. (2010b). Good question! Statistical ranking for question generation. In *Proc. of NAACL-HLT*.

Heilman, M. and Smith, N. A. (2010c). Rating computer-generated questions with Mechanical Turk. In *Proc. of the NAACL-HLT workshop on Creating Speech and Language Data With Amazon's Mechanical Turk*.

Heilman, M. and Smith, N. A. (2010d). Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proc. of NAACL*.

Heilman, M., Zhao, L., Pino, J., and Eskenazi, M. (2008). Retrieval of reading materials for vocabulary and reading practice. In *Proc. of the Third Workshop on Innovative Use of NLP for Building Educational Applications*.

Higgins, D. (2003). A machine-learning approach to the identification of WH gaps. In *Proc. of EACL*.

Hoffmann, R., Zhang, C., and Weld, D. S. (2010). Learning 5000 relational extractors. In *Proc. of ACL*.

Ide, N. and Suderman, K. (2004). The American National Corpus First Release. In *Proc. of LREC*.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proc. of KDD*.

Johnson, W. L., Rickel, J. W., and Lester, J. C. (2000). Animated pedagogical agents: Face-to-face interaction in interactive learning environments. *International Journal of Artificial Intelligence in Education*, 11.

Jurafsky, D. and Martin, J. H. (2008). *Speech and Language Processing*. Prentice Hall, 2nd edition.

Katz, G. and Giesbrecht, E. (2006). Automatic identification of non-compositional multi-word expressions using Latent Semantic Analysis. In *Proc. of the Workshop on Multiword Expressions*.

Keenan, J. M., Betjemann, R. S., and Olson, R. K. (2008). Reading comprehension tests vary in the skills they assess: Differential dependence on decoding and oral comprehension. *Scientific Studies of Reading*, 12(3).

Klein, D. and Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. In *Advances in NIPS 15*.

Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In *Proc. of IEEE Int. Conf. Acoustics, Speech and Signal Processing*.

Knight, K. and Marcu, D. (2000). Statistics-based summarization - step one: Sentence compression. In *Proc. of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*.

Koedinger, K. R., Aleven, V., Heffernan, N., McLaren, B., and Hockenberry, M. (2004). Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In *Proc. of Intelligent Tutoring Systems*.

Koedinger, K. R., Anderson, J. R., Hadley, W. H., and Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 127–133.

Kok, S. and Brockett, C. (2010). Hitting the right paraphrases in good time. In *Proc. of NAACL-HLT*.

Kunichika, H., Katayama, T., Hirashima, T., and Takeuchi, A. (2004). Automated question generation methods for intelligent English learning systems and its evaluation. In *Proc. of ICCE*.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33.

Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *Proc. of ACL*.

Langkilde-Geary, I. (2002). An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proc. of the Second International Natural Language Generation Conference*.

Leacock, C., Chodorow, M., Gamon, M., and Tetreault, J. (2010). *Automated Grammatical Error Detection for Language Learners*. Morgan and Claypool.

Lehnert, W., Cardie, C., Fisher, D., McCarthy, J., Riloff, E., and Soderland, S. (1992). Description of the CIRCUS system as used for MUC-4. In *Proc. of MUC-4*.

Lehnert, W. G. (1978). *The process of question-answering*. Erlbaum, Hillsdale, NJ.

Levinson, S. C. (1983). *Pragmatics*. Cambridge University Press, Cambridge, MA.

Levy, R. and Andrew, G. (2006). Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proc. of LREC*.

Lin, C. (2004). ROUGE: a package for automatic evaluation of summaries. In *Proc. of Workshop on Text Summarization*.

Litman, D. and Silliman, S. (2004). Itspoke: An intelligent tutoring spoken dialogue system. In *Companion Proc. of HLT/NAACL*.

Liu, M., Calvo, R. A., and Rus, V. (2009). Automatic question generation for literature review writing support. In *Proc. of AIED*.

MacCartney, B. (2009). *Natural language inference*. PhD thesis, Stanford University.

Malone, T. and Lepper, M. R. (1987). Making learning fun. *Aptitude, Learning and Instruction: Conative and Affective Process Analyses*.

Mannem, P., Prasad, R., and Joshi, A. (2010). Question generation from paragraphs at UPenn: QGSTEC system description. In *Proc. of the Third Workshop on Question Generation*.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19.

McCarthy, D., Koeling, R., and Carroll, J. (2004). Finding predominant senses in untagged text. In *Proc. of ACL*.

McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proc. of ACL*.

McLaren, B. M., Wegerif, R., Mikŝátko, J., Scheuer, O., Chamrada, M., and Mansour, N. (2009). Are your students working creatively together? automatically recognizing creative turns in student e-discussions. In *Proc. of AIED*.

Melville, H. (1851). *Moby Dick*. Harper and Brothers.

Metzler, D. and Croft, W. B. (2005). Analysis of statistical question classification for fact-based questions. *Information Retrieval*, 8.

Meurers, D., Ziai, R., Amaral, L., Boyd, A., Dimitrov, A., Metcalf, V., and Ott, N. (2010). Enhancing authentic web pages for language learners. In *Proc. of the 5th Workshop on Innovative Use of NLP for Building Educational Applications*.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. J. (1990). WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4).

Miltsakaki, E. and Troutt, A. (2008). Real time web text classification and analysis of reading difficulty. In *Proc. of the Third Workshop on Innovative Use of NLP for Building Educational Applications*.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proc. of ACL-IJCNLP*.

Mitchell, T. M., Betteridge, J., Carlson, A., Jr., E. R. H., and Wang, R. C. (2009). Populating the semantic web by macro-reading internet text. In *Proc. of the Eighth International Semantic Web Conference*.

Mitkov, R. and Ha, L. A. (2003). Computer-aided generation of multiple-choice tests. In *Proc. of the HLT-NAACL workshop on Building educational applications using natural language processing*.

Mitkov, R., Ha, L. A., and Karamanis, N. (2006). A computer-aided environment for generating multiple-choice test items. *Natural Language Engineering*, 12(2).

Mohler, M. and Mihalcea, R. (2009). Text-to-text semantic similarity for automatic short answer grading. In *Proc. of EACL*.

Mostow, J., Beck, J., Bey, J., Cuneo, A., Sison, J., Tobin, B., and Valeri, J. (2004). Using automated questions to assess reading comprehension, vocabulary, and effects of tutorial interventions. *Technology, Instruction, Cognition and Learning*, 2:97–134.

Mostow, J. and Chen, W. (2009). Generating instruction automatically for the reading strategy of self-questioning. In *Proc. of AIED*.

National Institute of Child Health and Human Development (2000). *Report of the National Reading Panel. Teaching children to read: An evidence-based assessment of the scientific research literature on reading and its implications for reading instruction (NIH Publication No. 00-4769)*. U.S. Government Printing Office, Washington, DC.

Nelder, J. and Wedderburn, R. (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3).

Nenkova, A. (2006). *Understanding the process of multi-document summarization: content selection, rewrite and evaluation*. PhD thesis, Columbia University.

Nenkova, A. (2008). Entity-driven rewrite for multi-document summarization. In *Proc. of IJCNLP*.

Nielsen, R. D., Ward, W., Martin, J. H., and Palmer, M. (2008). Extracting a representation from text for semantic analysis. In *Proc. of ACL-08:HLT*.

Nyberg, E. H. and Mitamura, T. (1992). The KANT system: fast, accurate, high-quality translation in practical domains. In *Proc. of COLING*.

Och, F. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1).

Oller, Jr., J. W. (1972). Scoring methods and difficulty levels for cloze tests of proficiency in english as a second language. *The Modern Language Journal*, 56(3).

Ozuru, Y., Rowe, M., O'Reilly, T., and McNamara, D. S. (2008). Where's the difficulty in standardized reading tests: The passage or the question? *Behavior Research Methods*, 40.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

Pereira, F. and Warren, D. (1986). Definite clause grammars for language analysis. *Readings in natural language processing*.

Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer.

Pino, J., M. Heilman, M., and Eskenazi, M. (2008). A selection strategy to improve cloze question quality. In *Proc. of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. Ninth International Conference on Intelligent Tutoring Systems*.

Piwek, P. and Stoyanchev, S. (2010). Question generation in the CODA project. In *Proc. of the Third Workshop on Question Generation*.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3).

Prasad, R., Dinesh, N., Lee, A., Miltsakaki, E., Robaldo, L., Joshi, A., and Webber, B. (2008). The Penn Discourse Treebank 2.0. In *Proc. of LREC*.

Prasad, R. and Joshi, A. (2008). A discourse-based approach to generating why-questions from texts. In *Proc. of the Workshop on the Question Generation Shared Task and Evaluation Challenge*.

Quirk, C. and Corston-Oliver, S. (2006). The impact of parse quality on syntactically-informed statistical machine translation. In *Proc. of EMNLP*.

Quirk, C., Menezes, A., and Cherry, C. (2005). Dependency treelet translation: Syntactically informed phrasal SMT. In *Proc. of ACL*.

Radford, A. (1988). *Transformational Grammar*. Cambridge University Press.

Rankin, E. F. and Culhane, J. W. (1969). Comparable cloze and multiple-choice comprehension test scores. *Journal of Reading*, 13(3).

Razzaq, L., Patvarczki, J., Almeida, S. F., Vartak, M., Feng, M., Heffernan, N. T., and Koedinger, K. R. (2009). The ASSISTment Builder: Supporting the life cycle of tutoring system content creation. *IEEE Transactions on Learning Technologies*, 2(2).

Redfield, D. L. and Rousseau, E. W. (1981). A meta-analysis of experimental research on teacher questioning behavior. *Review of Educational Research*, 51(2).

Reiter, E. and Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering*, 3(1).

Riedel, S., Yao, L., and McCallum, A. (2010). Modeling relations and their mentions without labeled text. In *Proc. of the European Conference on Machine Learning and Knowledge Discovery in Databases*.

Riloff, E. and Schmelzenbach, M. (1998). An empirical approach to conceptual case frame acquisition. In *Proc. of the Sixth Workshop on Very Large Corpora*.

Ritter, S. (1998). The authoring assistant. In *Proc. of Intelligent Tutoring Systems*.

Ross, J. R. (1967). *Constraints on Variables in Syntax*. PhD thesis, MIT, Cambridge, MA.

Rus, V. and Graessar, A., editors (2009). *The Question Generation Shared Task and Evaluation Challenge*. Available at `http://www.questiongeneration.org/`.

Rus, V., Wyse, B., Piwek, P., Lintean, M., and Stoyanchev, S. (2010). Overview of the first question generation shared task evaluation challenge. In *Proc. of the Third Workshop on Question Generation*.

Ryan, F. L. (1973). Differentiated effects of levels of questioning on student achievement. *The Journal of Experimental Education*, 41(3).

Sag, I. A., Baldwin, T., Bond, F., Copestake, A., and Flickinger, D. (2002). Multiword expressions: A pain in the neck for NLP. In *Proc. of CICLing*.

Samson, G. E., Strykowski, B., Weinstein, T., and Walberg, J. H. (1987). The effects of teacher questioning levels on student achievement: A quantitative synthesis. *Journal of Educational Research*, 80(5).

Schwarz, G. E. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6(2).

Sgall, P., Hajičová, E., and Benešová, E. (1973). *Topic, focus and generative semantics*. Scriptor Verlag.

Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In *Proc. of HLT-NAACL*.

Shen, L. and Joshi, A. K. (2005). Ranking and reranking with the perceptron. *Machine Learning*, 60.

Shermis, M. D. and Burstein, J. (2003). *Automated essay scoring: A cross-disciplinary perspective*. MIT Press.

Skory, A. and Eskenazi, M. (2010). Predicting cloze quality for vocabulary training. In *Proc. of NAACL-HLT Workshop on Innovative Use of NLP for Building Educational Applications*.

Smith, D. A. and Eisner, J. (2006). Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proc. of the HLT-NAACL Workshop on Statistical Machine Translation*.

Smith, H. J., Higgins, S., Wall, K., and Miller, J. (2005). Interactive whiteboards: boon or bandwagon? a critical review of the literature. *Journal of Computer Assisted Learning*.

Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 13(3).

Sobin, N. (1987). The variable status of comp-trace phenomena. *Natural Language and Linguistic Theory*, 5(1).

Sukkarieh, J. and Bolge, E. (2008). Leveraging C-rater's automated scoring capability for providing instructional feedback for short constructed responses. In *Proc. of Intelligent Tutoring Systems*.

Szabo, A. and Hastings, N. (2000). Using IT in the undergraduate classroom: should we replace the blackboard with PowerPoint? *Computers and Education*, 35(3).

Takahashi, D. (1993). Movement of Wh-phrases in Japanese. *Natural Language and Linguistic Theory*, 11(4).

Taylor, M., Guiver, J., Robertson, S., and Minka, T. (2008). SoftRank: optimizing non-smooth rank metrics. In *Proc. of the International Conference on Web Search and Web Data Mining*.

Taylor, W. (1953). Cloze procedure: A new tool for measuring readability. *Journalism Quarterly*, 30.

Toutanova, K., Brockett, C., Gamon, M., Jagarlamudi, J., Suzuki, H., and Vanderwende, L. (2007). The PYTHY summarization system: Microsoft Research at DUC 2007. In *Proc. of DUC*.

Trees, A. R. and Jackson, M. H. (2007). The learning environment in clicker classrooms: student processes of learning and involvement in large university-level courses using student response systems. *Learning, Media and Technology*, 32(1).

van Gog, T. and Paas, F. (2010). Instructional efficiency: Revisiting the original construct in educational research. *Educational Psychologist*, 43(1).

Vanlehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M. (2005). The andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, 15(3).

Varges, S. (2006). Overgeneration and ranking for spoken dialogue systems. In *Proc. of the Fourth International Natural Language Generation Conference*.

Voorhees, E. M. (2004). Overview of the TREC 2003 question answering track. In *Proc. of TREC 2003*.

Walker, M. A., Rambow, O., and Rogati, M. (2001). Spot: a trainable sentence planner. In *Proc. of NAACL*.

Walsh, J. A. and Sattes, E. D. (2004). *Quality Questioning: Research-Based Practice to Engage Every Learner*. Corwin Press.

Wang, M., Smith, N. A., and Mitamura, T. (2007). What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proc. of EMNLP-CoNLL*.

Willen, W. W. and Clegg, A. A. (1986). Effective questions and questioning: A research review. *Theory and Research in Social Education*, 14(2).

Winne, P. H. (1979). Experiments relating teachers' use of higher cognitive questions to student achievement. *Review of Educational Research*, 49.

Woodsend, K., Feng, Y., and Lapata, M. (2010). Title generation with quasi-synchronous grammar. In *Proc. of EMNLP*.

Woolf, B. P. (2008). *Building Intelligent Interactive Tutors: Student-centered strategies for revolutionizing e-learning*. Morgan Kaufmann.

Wu, D. (1997). Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

Wyse, B. and Piwek, P. (2009). Generating questions from OpenLearn study units. In *Proc. of the Second Workshop on Question Generation*.

Xu, J. and Li, H. (2007). Adarank: a boosting algorithm for information retrieval. In *Proc. of SIGIR*.

Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proc. of ACL*.

Yao, X. and Zhang, Y. (2010). Question generation with minimal recursion semantics. In *Proc. of the Third Workshop on Question Generation*.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proc. of ACL*.

Zhang, D. and Lee, W. S. (2003). Question classification using support vector machines. In *Proc. of SIGIR*.

Zhu, Z., Bernhard, D., and Gurevych, I. (2010). A monolingual tree-based translation model for sentence simplification. In *Proc. of COLING*.

Appendix A

Rules for Identifying Potential Answer Phrases

This appendix describes the set of eighteen `Tregex` (Levy and Andrew, 2006) tree searching expressions used in stage 2 (§3.3.1 of Chapter 3) to identify "NP," "PP," and "SBAR" nodes that should not serve as answer phrases due to WH-movement constraints and other conservative restrictions encoded in the system. This rule set was constructed from descriptions of WH-movement phenomena by Goldberg (2006, Chapter 7) and Radford (1988, Chapter 9). We developed the rule set by creating a test suite of examples adapted from these sources and then augmenting that test suite when we observed failures on texts used as development data. The rule set is an extended and revised version of the one described by Heilman and Smith (2009).

When a pattern matches and a particular node is identified as a disallowed answer phrase, the system marks that node by adding the prefix "`UNMOVABLE-`" to its label (e.g., "`UNMOVABLE-NP`"). Phrases marked as such are skipped over when the system selects potential answer phrases from the noun phrase ("`NP`"), prepositional phrase ("`PP`"), and subordinate clause ("`SBAR`") nodes in an input sentence (§3.3.5 of Chapter 3). The first sixteen patterns are applied essentially in parallel to identify various types of constructions. Then, as discussed at the end of this appendix, two patterns are used to propagate constraints down the parse tree. See the paper by Levy and Andrew (2006) or the `Tregex` distribution for more information about the `Tregex` syntax.

Rule 1: The following rule is used to mark clauses (i.e., "`S`" nodes) that are under verb phrases

and are signaled as adjuncts by being offset by commas. Note that this and the other constraints will be propagated down to potential answer phrases (i.e., "NP," "PP," and "SBAR" nodes) by the final two rules in the rule set. This constraint enables the system to avoid generating questions such as **What did James hurry, barely catching?* from the sentence *James hurried, barely catching the bus*.

```
VP < (S=unmv $,, /,/)
```

Rule 2: The following rule is used to mark clause-level modifiers (i.e., nodes directly under a clausal node labeled "S") other than nouns and verbs. As with the previous rule, this constraint will be propagated down the tree by later rules. It enables the system to avoid generating **What did before taking James studied?* from the sentence *Before taking the exam James studied*. Note that due to the conservative restriction in Rule 15, the system does not currently generate questions with answer phrases that are prepositional phrases with no noun phrase object (e.g., *When did James study?* from *before taking the exam*).

```
S < PP|ADJP|ADVP|S|SBAR=unmv > ROOT
```

Rule 3: The following rule is used to mark phrases under conjunctions. A single conjoined node cannot undergo WH-movement if its siblings do not (e.g., **Who did John meet and Mary?* from the sentence *John met Bob and Mary.*).

```
/\\.*/ < CC << NP|ADJP|VP|ADVP|PP=unmv
```

Rule 4: The following rule is used to mark noun phrases in adjunct clauses. It assumes that subordinate clauses with an explicit complementizer other than *that* are adjuncts, whose descendents should not be answer phrases (e.g., *the bus* in *before the bus left*). For example, it enables the system to avoid, for example, generating **What did James arrived before left?* from the sentence *James arrived before the bus left*.

```
SBAR < (IN|DT < /[^that]/) << NP|PP=unmv
```

Rule 5: The following rule is used to mark phrases under a question phrase. This constraint is related to the notion of subjacency (Chomsky, 1973), as is the next constraint. It enables the system

to avoid generating **What did Darwin study how evolve?* from the sentence *Darwin studied how species evolve*.

```
SBAR < /^WH.*P$/ << NP|ADJP|VP|ADVP|PP=unmv
```

Rule 6: The following rule is used to mark the subject of a complement phrase with an explicit complementizer. It helps the system avoid ungrammatical questions that might occur due to the complementizer-trace effect (Chomsky and Lasnik, 1977; Sobin, 1987). The system avoids, for example, generating **Who did John say that is old?* from the sentence *John said that Bob is old*.

```
SBAR <, IN|DT < (S < (NP=unmv !$,, VP))
```

Rule 7: The following rule is used to mark a node if it is under a clause that serves as a predicate of a larger clause with a copula verb. For example, it identifies the node for the *the park* in the sentence *John's favorite activity is to run in the park*. With this rule the system avoids generating **What is John's favorite activity to run in?*, for example.

```
S < (VP <+(VP)  (VB|VBD|VBN|VBZ <
 be|being|been|is|are|was|were|am)  <+(VP) (S <<
 NP|ADJP|VP|ADVP|PP=unmv))
```

Rule 8: The following rule is used to mark objects of prepositional phrases with prepositions other than *of* or *about*. The prepositions *of* and *about* usually signal that a prepositional phrase is a complement, while other prepositions signal that a prepositional phrase is an adjunct, whose descendents should not become answer phrases. Restricting the type of prepositional phrases in this way allows, for example, the question *What did John visit the capital of?* to be generated from *John visited the capital of Alaska*, but disallows the generation **What did John visit a city in?* from the sentence *John visited a city in Alaska* (some might consider the latter question to be grammatical).

```
NP << (PP=unmv !< (IN < of|about))
```

Rule 9: The following rule is used to mark prepositional phrases that are nested within other prepositional phrases. It disallows, for example, the question **What did Bill see John in the hall of?*

from the sentence *Bill saw John in the hall of mirrors*.

```
PP << PP=unmv
```

Rule 10: The following rule is used to mark prepositional phrases in subjects, since WH-movement cannot occur from within subjects. From the sentence *The capital of Russia is Moscow*, it disallows the movement of the phrase *Russia* and subsequent generation of the semantically incorrect question *What is the capital of Moscow?*, while still permitting appropriate questions such as *What is the capital of Russia?*.

```
NP $ VP << PP=unmv
```

Rule 11: The following rule is used to mark subordinate clauses that are not children of verbs. It disallows, for example, the generation of **What did John own a car?* from the subordinate clause *that was blue* in the sentence *James owned a car that was blue*, but still permits the generation of *What did James know?* from *James knew that John had arrived*.

```
SBAR=unmv [ !> VP | $-- /,/ | < RB ]
```

Rule 12: The following rule is used to mark subordinate clauses that are children of verbs but are not complements. The complementizers *how*, *whether*, and *that* under "IN" or "WHADVP" nodes signal that an "SBAR" node is a complement. Also, if a subordinate clause has a "WHNP" child, it is a complement. For example, the rule disallows the generation of $*$*What would John notice the problem?* from *John would notice the problem if someone told him about it*. However, the rule still permits the generation of *What would John notice?* from *John would notice that there was a problem*.

```
SBAR=unmv !< WHNP <
 (/^[^S].*/  !<< that|whether|how)
```

The next four rules are conservative restrictions that enable the current implementation to avoid special cases that it does not handle.

Rule 13: The following rule is used to mark existential *there* noun phrases. For example, it disallows the generation of $*$*What was a dog in the park?* from *There was a dog in the park*.

```
NP=unmv < EX
```

While existential *there* constructions are not handled in the current implementation, a potential stage 1 transformation could convert them into sentences with standard word ordering (e.g., *A dog was in the park*), facilitating conversion into questions (e.g., *What was in the park?*).

Rule 14: The following rule is used to mark phrases that occur within direct quotations.

```
/^S/ < `` << NP|ADJP|VP|ADVP|PP=unmv
```

Rule 15: The following rule is used to mark prepositional phrases that do not have a noun phrase object (e.g., *before taking the exam*, *by studying all night*). Future extensions to the system could enable generation from such phrases (e.g., generation of *when* and *how* questions).

```
PP=unmv !< NP
```

Rule 16: The following rule is used to mark both of a pair of noun phrases that are siblings, such as in double-object dative constructions and sentence with clause-level temporal modifiers. For examples, by using this rule, the system avoids generating **Who did John give a book?* from *John gave Mary a book*.

```
NP=unmv $ @NP
```

After the above rules are applied, the following two expressions are applied in order to to propagate the constraints down the trees.

Rule 17: The first is used to mark as unmovable all nodes that are are descendents of an otherwise *movable* node. It encodes that noun phrases are generally islands (i.e., their descendents cannot undergo movement) and also conservatively restricts movement from within other phrase types.

```
NP|PP|ADJP|ADVP << NP|ADJP|VP|ADVP|PP=unmv
```

Rule 18: The second is used to mark as unmovable all nodes that are descendents of an unmovable node. Note that these last two rules could be collapsed into one for conciseness.

```
@UNMV << NP|ADJP|VP|ADVP|PP=unmv
```

Appendix B

Instructions for Question Annotation

This appendix presents the instructions given to question annotators. The instructions were presented on a web page, with slightly different formatting.

Instructions

The purpose of this task is to rate questions according to their acceptability as assessment items for literal reading comprehension–that is, for basic understanding of the facts presented in a text and not necessarily the implications, mood, consequences, etc. The questions might be given to a reader either by themselves as short answer questions, or as part of multiple-choice items.

Please rate each question independently. That is, if two questions are very similar, please rate them both the same even though they may be redundant. Also, don't penalize questions too much for being very obvious. For example, if a question's answer happens to be the subject of title of the article, but the question is otherwise OK, then please rate the question as acceptable.

For each article, please briefly skim the text first before rating any questions, to get an idea of the subject matter (you don't need to spend more than a minute or so on this). Then, click on one of the question numbers above the text.

For each question, read the sentence from which it was generated—which will be highlighted in yellow—as well as the surrounding context. Then read the question and rate it by selecting one of the 5 available options (good, acceptable, borderline, unacceptable, bad). Please refer to the list of reasons that a question may be unacceptable as well as the examples (see below).

Questions that have not been rated will have black numbers in the list of questions to rate. Questions that have been completed will have green numbers (Note: you don't have to click on any sort of "submit" button). Please rate all the questions for each article in the list of assigned articles (i.e., make sure all the numbers turn green).

If you are not a native speaker of English, please do not participate.

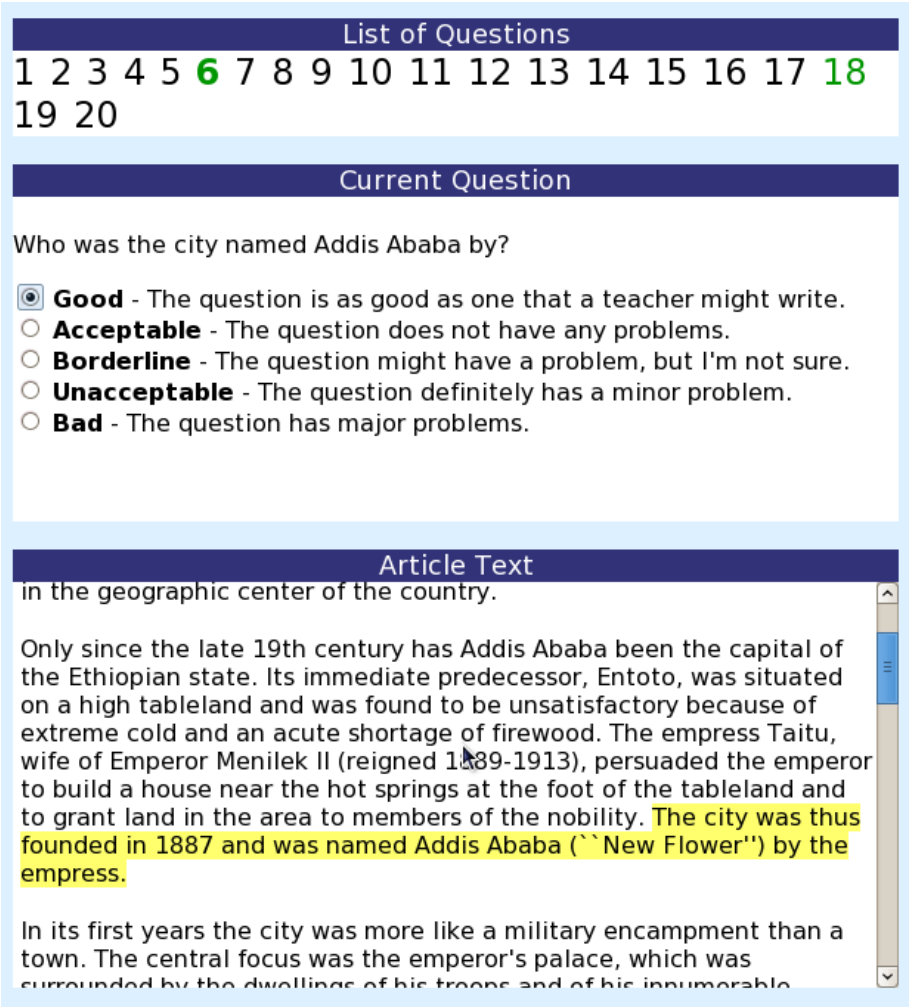

Figure B.1: A screenshot of the question annotation interface.

(Un)grammaticality	The question is ungrammatical, does not make sense, or uses the wrong question word (e.g., who, what, which, etc.).
Incorrect Information	The question implies something that is obviously incorrect, according to the given context.
Vagueness	The question has no simple and clear answer (even a good reader would not know the answer).
Awkwardness/Other	The question is very awkwardly phrased., or has some other problem (e.g., no native speaker of English would say it this way).

Table B.1: The list of reasons that questions should be annotated as unacceptable.

Rating	**Example Question**	**Explanation**
Bad	What also might have?	It is so vague that it could not easily be revised to make a good question.
Bad	The British evacuated who moved his army to New York City?	It does not make any sense.
Unacceptable	In 1978, what was awarded the Nobel Prize in Economics?	”What” should be ”who.” (This can be fixed fairly easily, though.)
Bad/Unacceptable	Who reorganized the army during the standoff?	Vague. What ”the army” and ”the standoff” refer to is not clear.
Acceptable	What is the traditional religion of Japan?	It has no apparent problems.
Good/Acceptable	Who was deprived of both the knighthood and earldom after taking part in the Jacobite rising of 1715?	It has no problems and asks about complex information.

Table B.2: Some examples of annotated questions.

Appendix C

User Study Instructions

The instructions given to participants in user study described in Chapter 5 were as follows. Note that the instructions were presented on a web page with different formatting.

The software tool being tested in this experiment aims to help educators create reading quizzes for new texts (e.g., from the web). The tool displays ranked lists of candidate questions, from which you can select and revise questions that you think are acceptable.

This study will test whether users can more efficiently create factual questions using the tool than when they have to write them on their own. During the study, you will use two versions of the tool. In one version, the tool will automatically generate and suggest questions, and you will be able to select and revise the suggested questions or write your own. In the other version of the tool, you will be asked to write all the questions on your own. In both versions, please try to create the questions quickly (without sacrificing quality).

Here are a few things to keep in mind.

Factual Questions

The tool focuses on recognition and recall questions, and on the literal information in a text. Please create questions that assess factual knowledge (e.g. who did something, when things happened, where things happened, what something is, etc.).

Please try to avoid questions that would require high-level reasoning or complex inference (e.g., why things happened, what will happen next, etc.).

Imagine that you are writing only *part* of a reading quiz. The part you are writing would as-

sess whether students read the text closely enough to remember basic factual information (students wouldn't be able to re-read the text while taking the quiz). A separate part of your quiz (which we won't deal with in this study) would assess whether students have a deeper understanding of the text.

Here is an example:

Text:

- . . . In 1978, Herbert Simon was awarded the Nobel Prize in Economics . . .

Factual questions (please create these sorts of questions):

- When was Herbert Simon awarded the Nobel Prize in Economics?
- Who was awarded the Nobel Prize in Economics in 1978?

Some deeper questions (please avoid these sorts of questions):

- What was Herbert Simon's area of scientific expertise?
- Was Herbert Simon respected in his field?

Correcting Errors

The tool makes many errors. Please make sure that the suggestions that you select from the tool are grammatically well-formed, not vague, etc. If the tool suggests a question that you like, and if you notice that the question has a minor error, please make sure to fix the error. If the tool does not suggest enough good questions, you may also write your own.

Here is an example:

A question with an error:

- In 1978, *what* was awarded the Nobel Prize in Economics?

A revised question:

- In 1978, *who* was awarded the Nobel Prize in Economics?

Answers

Please make sure to add or revise the answers to the questions that you create. For yes-no questions, you can just type in "yes" or "no."

The instructions also included the labeled interface screenshot in Figure 5.1 in Chapter 5. Also, specific prompts (e.g., "Please select or create 3 factual questions about the article below") were provided before each article.

Appendix D

Alternative Ranking Methods

In this section, we describe in more detail the three ranking methods used in the experiments in §4.7 of Chapter 4. The three methods are maximum margin-based models trained using online passive aggressive (PA) learning (Crammer et al., 2006). PA learning and the closely related MIRA algorithm (Crammer and Singer, 2003) have been shown to lead to excellent performance on other NLP tasks (McDonald et al., 2005; Chiang et al., 2008).

Here, we describe a PA version of linear regression for ranking, a ranker based on binary classification, and a pairwise ranker. These rankers can be viewed as instances of a general framework for PA learning, based on work by Crammer et al. (2006), which we discuss in this section. The rankers are distinguished by particular representations, loss functions, and prediction rules.

Regression

The first alternative ranking method is an online PA version of linear regression similar to support vector regression (Smola and Schölkopf, 2004). The pseudocode, shown in Algorithm 3, is adapted from Crammer et al. (2006).

Algorithm 3 `PassiveAggressiveLinearRegression`$((\mathbf{X}, \mathbf{Y}), C, T)$:

A passive aggressive learning algorithm for linear regression, following Crammer et al. (2006). H_t, ℓ_t, and G_t can be modified for binary classification or pairwise ranking (see text).

$\mathbf{w}_1 = (0, \ldots, 0)$

$\mathbf{w}_{avg} = (0, \ldots, 0)$

for $t = 1, 2, \ldots, T$ **do**

sample: $(x_t, y_t) \in (\mathbf{X}, \mathbf{Y})$

compute prediction for sample: $H_t = \mathbf{w}_t^\top \mathbf{f}(x_t)$

compute loss based on prediction: $\ell_t = \max\left\{0, |\mathbf{w}_t^\top \mathbf{f}(x_t) - y_t| - \epsilon\right\}$

compute update based on loss: $G_t = \min\left\{C, \frac{\ell_t}{\|\mathbf{f}(x_t)\|^2}\right\} \text{sign}(y_t - H_t)\mathbf{f}(x_t)$

update parameters: $\mathbf{w}_{t+1} = \mathbf{w}_t + G_t$

update average: $\mathbf{w}_{avg} = \mathbf{w}_{avg} + \mathbf{w}_{t+1}/T$

end for

return $\mathbf{w}_{avg}$

The range of the output is the set of real numbers ($\mathcal{Y} = \mathbb{R}$). The loss function, the ϵ-insensitive loss,[1] returns 0 for predictions H_t within a small value (ϵ) of the target $\mathbf{y}_t$, and then increases linearly beyond that. Typically, ϵ is set to some small, positive value (we set $\epsilon = 0.001$ in our experiments). Support vector regression (Smola and Schölkopf, 2004) optimizes the same loss function.

Note that the equations in Algorithm 3 do not include an intercept parameter. Instead, they only model the slope. To maintain simplicity, an explicit intercept can be avoided either by normalizing the data to have zero mean, as well as recording the original mean and standard deviation, or, as we do, by including a special bias feature that always has value 1 in the output of the feature function (f).

Algorithm 3 performs averaging over the parameters from each iteration rather than returning the final parameter vector. Averaging has been shown theoretically and empirically to improve performance in online learning (Freund and Schapire, 1999; Collins, 2002).

The aggressiveness parameter C allows the algorithm to better handle non-separable data. It is

[1] One could also theoretically use the mean-squared error as in ordinary least square linear regression (which has a closed form solution).

analogous to the complexity parameter in a support vector machine (and is similarly derived by introducing slack variables). If C is positive infinity, the updated parameters $\mathbf{w}_{t+1}$ will guarantee that the current example x_t will be labeled with zero loss. Otherwise, updates will often be smaller in order to retain the information encoded in the existing set of parameters. Typically, C is set to a value around 1, and performance does not usually depend heavily on the particular setting used.[2]

The "sample" step for selecting a training example pair (x_t, y_t) can be implemented in various ways. One can randomly sample from the training data $(\mathbf{X}, \mathbf{Y})$ at each iteration t. Or, to ensure that all instances are included, one can iterate through the list $(\mathbf{X}, \mathbf{Y})$, shuffle it once the end is reached, iterate through again, and so on until T examples have been processed. We take the latter approach of shuffling and iterating in our experiments.[3]

Binary Classification

The second alternative method ranks questions by sorting the scores from a binary classifier for whether a question is acceptable or not. The output can take two values: $\mathcal{Y} = \{-1, +1\}$, where $y = -1$ denotes an unacceptable question and $y = 1$ denotes an acceptable question. To train this ranking model, we map 1 to 5 question acceptability ratings to binary values that indicate whether the threshold of 3.5 is exceeded. To rank new questions, we simply sort by their predicted real-valued scores for the "acceptable" class. Following Crammer et al. (2006), we can modify Algorithm 3 for binary classification as shown below.

$$H_t = \text{sign}(\mathbf{w}_t^\top \mathbf{f}(x_t)) \tag{D.1}$$

$$G_t = \min \left\{ C, \frac{\ell_t}{\|\mathbf{f}(x_t)\|^2} \right\} y_t \mathbf{f}(x_t) \tag{D.2}$$

$$\ell_t = \max\{0, 1 - y_t \mathbf{w}_t^\top \mathbf{f}(x_t)\} \tag{D.3}$$

[2] In our experiments, we set $C = 1.0$ and $T = 10$.

[3] What we present here is what Crammer et al. (2006) refer to as "PA-I" (e.g., in Figure 1 of that paper). They also introduce two other variants.

Pairwise Ranking

The third alternative ranking method is a pairwise ranking model. Pairwise ranking models, such as SVM$_{rank}$ (Joachims, 2002) and the ranking perceptron (Gao et al., 2005; Shen and Joshi, 2005), have been popular for creating ranked lists of retrieved documents in information retrieval applications.

Given a list of items as input, we aim to predict a list of indices that define a ranking of those items (e.g., a ranking of web documents for a set of queries). Following Gao et al. (2005), the training input consists of all pairs of items that have different values (i.e., for their mean acceptability ratings in our QG application). Also following Gao et al. (2005), the set of pairs that the algorithm considers is restricted to pairs of items that belong to the same group. For text retrieval, these groups are queries and the items are documents that may be relevant to the query. For QG, the groups are input texts and the items are questions about those texts.

We aim to rank each of the higher-valued items above each of the lower-valued items. Moreover, we seek a model that separates items with different values by a large margin, and so we adapt the hinge loss for the loss function. The pairwise ranking approach can handle items with binary, ordinal, or continuous labels—or even pairwise preferences such as in clickthrough data, as shown by Joachims (2002). The inputs x_t are pairs of items with different values,[4] and we denote $x_{t,1}$ and $x_{t,2}$ to be the first and second items in the pair, respectively. y_t has value 1 when the first item should be ranked above the second, and -1 when the second should be ranked above the first.

PA for pairwise ranking can be instantiated by modifying Algorithm 3 to use the equations below.

$$H_t = \text{sign}(\mathbf{w}_t^\top (\mathbf{f}(x_{t,1}) - \mathbf{f}(x_{t,2}))) \tag{D.4}$$

$$G_t = \min \left\{ C, \frac{\ell_t}{\|\mathbf{f}(x_{t,1} - x_{t,2})\|^2} \right\} y_t (\mathbf{f}(x_{t,1}) - \mathbf{f}(x_{t,2})) \tag{D.5}$$

[4]Note that pairs of items with the same value are typically skipped. E.g., in text retrieval with binary relevance judgments, only pairs consisting of one relevant and one irrelevant document are considered (Gao et al., 2005).

$$\ell_t = \max\{0, 1 - y_t \mathbf{w}_t^\top (\mathbf{f}(x_{t,1}) - \mathbf{f}(x_{t,2}))\} \tag{D.6}$$

Once the model is trained to find weights $\hat{\mathbf{w}}$, a set of new items can be ranked by scoring each item $\mathbf{x}_i$ according to $\hat{\mathbf{w}}^\top \mathbf{f}(x_i)$ and then sorting by score.

Testing on a Standard Ranking Task

We tested our implementation of the pairwise PA ranking method using a standard dataset for learning to rank for text retrieval, and we found its performance to be quite similar to SVM$_{rank}$.

We used the standard training, development, and testing sets for the supervised learning portion of the LETOR 4.0 dataset from Microsoft Research.[5] This large-scale text retrieval dataset includes thousands of queries, along with feature vector representations of documents labeled by relevance.[6]

We used the standard evaluation metrics for this task: mean average precision (MAP) and normalized cumulative discounted gain (NDCG), both of which focus on good performance at the top of the rankings. For details on these metrics, see, for example, Chakrabarti et al. (2008).

Performance for this task is evaluated using 5-fold cross-validation. For each iteration of cross-validation, there is a training, development, and testing set. We tuned hyperparameters (e.g., the aggressiveness parameter) with the development set. We present results for the testing set, averaged across the 5 folds. The 5 folds also provide us with a measure of the variance of ranking performance, which allows us to test statistical significance with a paired t-test.

The LETOR release also include previously reported results for a few strong baselines, including SVM$_{rank}$ (Joachims, 2002), a max-margin pairwise ranking model very similar to the PA pairwise model, and the state-of-the-art AdaRank$_{NDCG}$ algorithm (Xu and Li, 2007), a boosting method that directly optimizes the NDCG metric.

The results, presented in Table D.1, show that the pairwise ranking model's performance is not significantly different from SVM$_{rank}$ ($p > 0.05$, paired t-test). The PA linear regression provides performance that is significantly worse than the other models, which is expected since the data are

[5] The LETOR datasets are available at `http://research.microsoft.com/`

[6] We did not test the PA binary classifier implementation because the relevance judgments in the LETOR dataset are ternary (i.e., 0, 1, or 2).

	MAP		NDCG		
Algorithm	**Mean**	**S.D.**	**Mean**	**S.D.**	**Settings**
PA linear	.4480 $^{\#}$	.0480	.4642 $^{*,\#}$	.0472	C=0.1, T=10, ϵ=0.5
PA pairwise	.4693 $^{\#}$	.0412	.4867	.0482	C=0.1, T=10
SVM$_{rank}$	.4696	.0421	.4832	.0462	-
AdaRank$_{NDCG}$	.4824	.0436	.4950	.0472	-

Table D.1: Means and standard deviations for performance of the passive aggressive algorithms at the task of learning to rank for text retrieval, with previously reported results for comparison. * indicates that a mean performance value is significantly different from the mean performance of SVM$_{rank}$ ($p < 0.01$, 2-tailed paired t-test). # indicates that performance is significantly different from AdaRank$_{NDCG}$ ($p < 0.01$, paired t-test).

not on a continuous scale. The more sophisticated AdaRank$_{NDCG}$ significantly outperforms all other methods.

It is worth noting that, in all cases, the differences in performance are only a few points of MAP or NDCG. The extent to which these small differences in ranking performance affect the experience of end users of a system is questionable—a simple ranking approach may often suffice.

Appendix E

Sentence Simplification Example from *Moby Dick*

An important challenge in QG is the extraction of information from complex sentences, as discussed in §2.2.2 of Chapter 2. As an extreme example of a complex sentence that we might want to simplify in order to extract information from, consider the following 133-word sentence in example E.1 provided by Melville (1851, Chapter 96).

(E.1) As they narrated to each other their unholy adventures, their tales of terror told in words of mirth; as their uncivilized laughter forked upwards out of them, like the flames from the furnace; as to and fro, in their front, the harpooneers wildly gesticulated with their huge pronged forks and dippers; as the wind howled on, and the sea leaped, and the ship groaned and dived, and yet steadfastly shot her red hell further and further into the blackness of the sea and the night, and scornfully champed the white bone in her mouth, and viciously spat round her on all sides; then the rushing Pequod, freighted with savages, and laden with fire, and burning a corpse, and plunging into that blackness of darkness, seemed the material counterpart of her monomaniac commander's soul.

A QG system would have great difficulty with such a sentence: one cannot generate 100+ word questions. In order to generate reasonably concise questions, one must break down such an input, either explicitly or implicitly, into parts that can be processed more easily.

From a gold-standard parse of sentence E.1 created by the author, the simplified factual statement extractor discussed in §3.2.1 can extract the following 16 much simpler sentences:

- The rushing Pequod seemed the material counterpart of her monomaniac commander's soul.
- They narrated to each other their unholy adventures.
- Their uncivilized laughter forked upwards out of them.
- The harpooneers wildly gesticulated with their huge pronged forks and dippers in their front.
- The wind howled on.
- The sea leaped.
- The ship groaned.
- The ship dived.
- The ship steadfastly shot her red hell further and further into the blackness of the sea and the night.
- The ship scornfully champed the white bone in her mouth.
- The ship viciously spat round her on all sides.
- The rushing Pequod was freighted with savages.
- The rushing Pequod was laden with fire.
- The rushing Pequod was burning a corpse.
- The rushing Pequod was plunging into that blackness of darkness.
- Their unholy adventures were their tales of terror told in words of mirth.

Of course, such extremely long sentences are not a major concern for most NLP applications (including our QG application), but this example shows that our system is capable of processing very complex inputs with constructions embedded within multiple levels of other constructions. Also, it is worth noting that automatic parses for such complex sentences usually contain many errors, which will propagate into errors in the extracted outputs.[1]

[1] If the parse from the Stanford Parser is passed as input, which contains numerous errors, the system does not extract high-quality outputs. The parser also required 4 GB of memory and spent 10 minutes processing the original sentence.