Question Generation as a Competitive Undergraduate Course Project

Noah A. Smith
Language Technologies Inst.
Carnegie Mellon University
Pittsburgh, PA 15213, USA
nasmith@cs.cmu.edu

Michael Heilman
Language Technologies Inst.
Carnegie Mellon University
Pittsburgh, PA 15213, USA
mheilman@cs.cmu.edu

Rebecca Hwa
Computer Science Dept.
University of Pittsburgh
Pittsburgh, PA 15260, USA
hwa@cs.pitt.edu

Abstract

We describe a competitive question generation and answering project used in our undergraduate natural language processing courses. This semester-long project challenges teams of three or four students to use available NLP components (or develop their own) to construct systems that ask and answer questions about an arbitrary Wikipedia article. We describe how the project and competition were structured, the outcomes, and lessons learned.

1 Introduction

Question generation and question answering are two key challenges for natural language understanding and interfaces (Voorhees, 2004; Kunichika et al., 2004; Mitkov et al., 2006). While question generation and answering have clear practical applications, we believe they also have high pedagogical value. The techniques taught in natural language processing (NLP) courses—language modeling, part-of-speech tagging, named entity recognition, parsing, etc.—are directly applicable, so existing NLP components can be used to aid development, familiarizing students with NLP tools without requiring everything to be built from scratch.

Unlike translation, system inputs and outputs are easy to produce on the fly and both are easily evaluated by NLP novices. Crucially, good performance does not appear to rely on large-scale system engineering or the use of massive amounts of data, so it is possible for a small group of students to build a working system inside a semester.

The two problems together provide an excellent opportunity to teach students not only about the challenges, sub-problems, and techniques of NLP, but also about annotation, inter-annotator agreement, and evaluation.

In this paper, we describe a semester-long, undergraduate course project built on question generation and answering. This project was used in 2008 in our introductory NLP courses.

2 Project Instructions

The project requirements provided considerable flexibility. Students could develop their systems in any programming language, and they were allowed to use existing NLP components available on the Web. The command-line interface for the question generation program was

```
./ask art.txt N
```

where `art.txt` is a file containing the text of a Wikipedia article, and `N` is a positive integer telling how many questions to generate. The program is expected to print to standard output a sequence of newline-separated `N` questions about the article that a human could answer, given the article. Students were instructed to aim for questions that are "**fluent** and **reasonable**."

The answering program has a similar interface:

```
./answer art.txt q.txt
```

where `q.txt` lists questions in the same format as `ask`'s output. Answers are to be written to standard output, one per line. Students were instructed to aim for answers that are "**fluent**, **correct**, and **intelligent**."

Note that there is no document retrieval component to this project; questions and answers always pertain to a specific, known document.

3 Timeline

The project proceeded in 4 phases of a 15-week semester: **data preparation** (weeks 1–4), during

which the first few course lectures introduced the most important concepts for getting started in NLP and motivating applications; **system development** (weeks 5–12), during which teams worked on their systems as they learned more about problems and solutions in NLP; **evaluation/competition** (weeks 13–14); and **live demonstrations** (hosted by the local Google office) at the end. The first and third phases are most relevant.

Data Preparation An early assignment asked students to produce questions about two Wikipedia articles, 3 easy, 3 medium, and 3 hard.[1] A subsequent homework asked each student to rate a set of questions generated by other students to ensure that they were neither "too easy" nor "too hard," given the relevant article. Students could also indicate if questions were otherwise unacceptable. Each student then provided answers for all of the acceptable questions. After teams were formed, each team was given the union of valid questions its members had generated or answered, as development data.

Evaluation The full set of 542 questions generated in the first phase was used to evaluate the systems at the end of the project. In addition, each team's `ask` program was used to generate an additional set of 120 questions (3 for each of 40 articles). The first round of evaluation asked the students to judge and answer these system-generated questions. If a question was disfluent or unreasonable, it was to be marked as such (and would subsequently not be used in the evaluation).

Out of the 558 valid human-generated and 651 system-generated questions, a sample of 381 questions was given to the `answer` programs, and the second round of evaluation asked students to rate the correctness and fluency of answers.[2] To illustrate the gap between system and human performance, the human questions were included in the evaluation.[3]

[1] Space does not permit inclusion of the guidelines.

[2] A sample, stratified by article, was taken because the students could not feasibly evaluate all answers for the whole set. The size is due to a lack of acceptable questions for a few articles

[3] The evaluation is not perfectly rigorous. Each team had access to a portion of the human-generated evaluation questions throughout the exercise. Second, we took no measures to prevent students from cheating at evaluation by determining their own output and inflating the scores.

4 Outcomes

Student-Developed Systems Most of the student-developed systems had the following characteristics in common.

- Multiple levels of preprocessing of the input articles using existing, freely available NLP tools for various tasks such as sentence boundary detection, tokenization, parsing, named entity recognition, and coreference resolution.
- Use of existing knowledge sources, in particular the WordNet lexical database to replace words with synonyms, antonyms, and other related words.
- Hand-written transformation rules for generating questions from source text. These rules operated on either lists of words, dependency parse trees, or phrase-structure parse trees.
- Coverage of only a few of the question types and linguistic phenomena related to questions. Most groups chose to focus on particular cases in which questions were easy to generate (e.g., source sentences consisting of just a noun phrase followed by a verb phrase).

Individual teams also devised unique strategies for particular challenges. Several teams judged that the tools available for coreference and named entity recognition were difficult to incorporate or inaccurate. One group addressed this issue by implementing a simplistic coreference resolver that replaced anaphora, such as "it" or "he", with the nearest previous noun phrase mention of the appropriate type. Another group, in order to identify when to use "who" rather than "what" at the beginning of a question, used a list of common names from U.S. Census data.

One team developed a novel approach to authoring the patterns used by their system to transform declarative sentences into questions. They created a web interface for rating the questions that were generated from an initial set of patterns. Team members and their friends rated the questions as good or bad, and those ratings were used to identify which patterns were most useful and which needed to be revised and re-evaluated.

team:	1	2	3	4	5	6	7	8	h.
fraction of questions judged to have high quality:	.68	.69	–	.69	.74	.53	.62	.30	.92
fraction of answers judged to have high quality:	.24	.61	.45	.41	.53	.44	.61	.06	.89

Table 1: Performance of the 8 teams, on asking and answering. Column "h." shows human-authored items. Team 3 failed to produce a question generator.

Another group's system ranked sentences from the source article according to a team-developed measure of the well-definedness of context. The aim of this strategy was to generate more fluent questions by using the top-ranked sentences with less ambiguity (e.g., fewer anaphora).

Evaluation Results Table 1 shows the results of the final evaluation of each team's questions and answers.

Some system-generated questions were well-formed and challenging. For example, from the Wikipedia article about penguins, a system generated the question, "What happens when mothers lose a chick?" Other examples of system-generated output illustrate challenges faced in question generation. Many questions were overly vague, e.g., "Is it true that [U.S. President Grover C]leveland referred the matter to [C]ongress?" Syntactic violations were not uncommon, e.g., "Was Qatar taboo for men to wear shorts in public?"

Many of the system-generated questions were Yes/No questions, and most of them had "Yes" as the correct answer. These can often be extracted easily from the text, in contrast to questions with "No" as the correct answer.

Informal feedback from the students was generally positive, noting that the project was challenging and more open-ended than usual but satisfying, particularly with the competitive element.

5 Lessons Learned

Although most teams' questions scored higher than their answers, we do not take this to mean the asking task is easier. Our difficulty guidelines led students to avoid complex questions requiring inference based on multiple propositions. The design of the project encouraged students to try to "trick" other teams' systems by forming reasonable, even easy questions, via linguistic transformations. Yet while some teams moved in this direction, most found that currently available core NLP components for anaphor resolution, synonyms, etc., were either too difficult to use, too slow, or insufficiently accurate. The resulting question generation systems typically involved hand-crafted rules.

We close by noting the challenge of evaluation. In the closed world of a classroom, system builders can be coerced into preparing and evaluating data (a useful exercise, we argue). But time limitations and lack of annotation expertise, as well as real differences in people's comprehension of articles they read, lead us to believe that inter-rater agreement might have been quite low. And of course the usual difficulties of defining unambiguous guidelines for difficulty and fluency are even more pronounced when participants are students only just becoming comfortable with the idea that—in contrast to most of computer science—"correct system output" is in the eye of the judge.

Acknowledgments

We gratefully acknowledge the efforts of the students who participated in this exercise, and Google for hosting the project demonstrations and sponsoring prizes. The authors were supported in part by NSF IIS-0713265 (to Smith), an NSF Graduate Research Fellowship (to Heilman), NSF IIS-0712810 and IIS-0745914 (to Hwa), and Institute of Education Sciences, U.S. Department of Education R305B040063 (to Carnegie Mellon).

References

H. Kunichika, T. Katayama, T. Hirashima, and A. Takeuchi. 2004. Automated question generation methods for intelligent English learning systems and its evaluation. In *Proc. of ICCE*.

R. Mitkov, L. An Ha, and N. Karamanis. 2006. A computer-aided environment for generating multiple-choice test items. *Nat. Lang. Eng.*, 12(2):177–194.

E. M. Voorhees. 2004. Overview of the TREC 2003 question answering track. In *Proc. of TREC 2003*.