Mind the Gap: Learning to Choose Gaps for Question Generation

Lee Becker
Department of Computer Science
University of Colorado Boulder
Boulder, CO 80309
lee.becker@colorado.edu

Sumit Basu and Lucy Vanderwende
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{sumitb,lucyv}@microsoft.com

Abstract

Not all learning takes place in an educational setting: more and more self-motivated learners are turning to on-line text to learn about new topics. Our goal is to provide such learners with the well-known benefits of testing by automatically generating quiz questions for on-line text. Prior work on question generation has focused on the grammaticality of generated questions and generating effective multiple-choice distractors for individual question targets, both key parts of this problem. Our work focuses on the complementary aspect of determining what part of a sentence we should be asking about in the first place; we call this "gap selection." We address this problem by asking human judges about the quality of questions generated from a Wikipedia-based corpus, and then training a model to effectively replicate these judgments. Our data shows that good gaps are of variable length and span all semantic roles, i.e., nouns as well as verbs, and that a majority of good questions do not focus on named entities. Our resulting system can generate fill-in-the-blank (cloze) questions from generic source materials.

1 Introduction

Assessment is a fundamental part of teaching, both to measure a student's mastery of the material and to identify areas where she may need reinforcement or additional instruction. Assessment has also been shown an important part of learning, as testing assists retention and can be used to guide learning (Anderson & Biddle, 1975). Thus, as learners move on from an educational setting to unstructured self-learning settings, they would still benefit from having the means for assessment available. Even in traditional educational settings, there is a need for automated test generation, as teachers want multiple tests for topics to give to different students, and students want different tests with which to study and practice the material.

One possible solution to providing quizzes for new source material is the automatic generation of questions. This is a task the NLP community has already embraced, and significant progress has been made in recent years with the introduction of a shared task (Rus et al., 2010). However, thus far the research community has focused on the problem of generating grammatical questions (as in Heilman and Smith (2010a)) or generating effective distractors for multiple-choice questions (Agarwal and Mannem, 2011).

While both of these research threads are of critical importance, there is another key issue that must be addressed – which questions should we be asking in the first place? We have highlighted this aspect of the problem in the past (see Vanderwende (2008)) and begin to address it in this work, postulating that we can both collect human judgments on what makes a good question and train a machine learning model that can replicate these judgments. The resulting learned model can then be applied to new material for automated question generation. We see this effort as complementary to the earlier progress.

In our approach, we factor the problem of generating good questions into two parts: first, the selection of sentences to ask about, and second, the identification of which part of the resulting sentences the question should address. Because we want to focus on these aspects of the problem and not the surface form of the questions, we have chosen to generate simple gap-fill (cloze) questions,

though our results can also be used to trigger Wh-questions or multiple-choice questions by leveraging prior work. For sentence selection, we turn to methods in summarization and use the simple but effective SumBasic (Nenkova et al., 2006) algorithm to prioritize and choose important sentences from the article. We cast the second part, gap selection, as a learning problem. To do this, we first select a corpus of sentences from a very general body of instructional material (a range of popular topics from Wikipedia). We then generate a constrained subset of all possible gaps via NLP heuristics, and pair each gap with a broad variety of features pertaining to how it was generated. We then solicit a large number of human judgments via crowdsourcing to help us rate the quality of various gaps. With that data in hand, we train a machine learning model to replicate these judgments. The results are promising, with one possible operating point producing a true positive rate of 83% with a corresponding false positive rate of 19% (83% of the possible *Good* gaps are kept, and 19% of the not-*Good* gaps are incorrectly marked); see Figure 6 for the full ROC curve and Section 4 for an explanation of the labels. As the final model has only minimal dependence on Wikipedia-specific features, we expect that it can be applied to an even wider variety of material (blogs, news articles, health sites, etc.).

2 Background and Related Work

There already exists a large body of work in automatic question generation (QG) for educational purposes dating back to the Autoquest system (Wolfe, 1976), which used an entirely syntactic approach to generate Wh-Questions from individual sentences. In addition to Autoquest, several others have created systems for Wh-question generation using approaches including transformation rules (Mitkov and Ha, 2003), template-based generation (Chen et al., 2009; Curto et al., 2011), and overgenerate-and-rank (Heilman and Smith, 2010a). The work in this area has largely focused on the surface form of the questions, with an emphasis on grammaticality.

Alternatively, generation of gap-fill style questions (a.k.a. cloze questions) avoids these issues of grammaticality by blanking out words or spans in a known good sentence. There is a large body of existing work that has focused on generation of this type of question, most of which has focused on vocabulary and language learning. The recent work of Agarwal and Mannem (2011) is closer to our purposes; they generated fill-in-the-blank questions and distractor answers for reading comprehension tests using heuristic scoring measures and a small evaluation set. Our work has similar aims but employs a data-driven approach.

The Question-Generation Shared Task and Evaluation Challenge (QG-STEC) (Rus et al., 2010) marks a first attempt at creating a common task and corpus for empirical evaluation of question generation components. However, evaluation in this task was manual and the number of instances in both the development and training set were small. As there exists no other dataset for question generation, we created a new corpus using Amazon Mechanical Turk by soliciting judgments from non-experts. Snow et al. (2008) have validated AMT as a valid data source by comparing non-expert with gold-standard expert judgments. Corpus creation using AMT has numerous precedents now; see i.e. Callison-Burch and Dredze (2010) and Heilman and Smith (2010b). We have made our corpus (see Section 4) available online to enable others to continue research on the gap-selection problem we address here.

3 Question Generation

To achieve our goal of selecting better gap-fill questions, we have broken down the task into stages similar to those proposed by Nielsen (2008): 1) sentence selection, 2) question construction, and 3) classification/scoring. Specifically, we utilize summarization to identify key sentences from a passage. We then apply semantic and syntactic constraints to construct multiple candidate question/answer pairs from a given source sentence. Lastly we extract features from these hypotheses for use with a question quality classification model. To train this final question-scoring component, we made use of crowdsourcing to collect ratings for a corpus of candidate questions. While this pipeline currently produces gap-fill questions, we envision these components can be used as input for more complex surface generation such as Wh-forms or distractor selection.

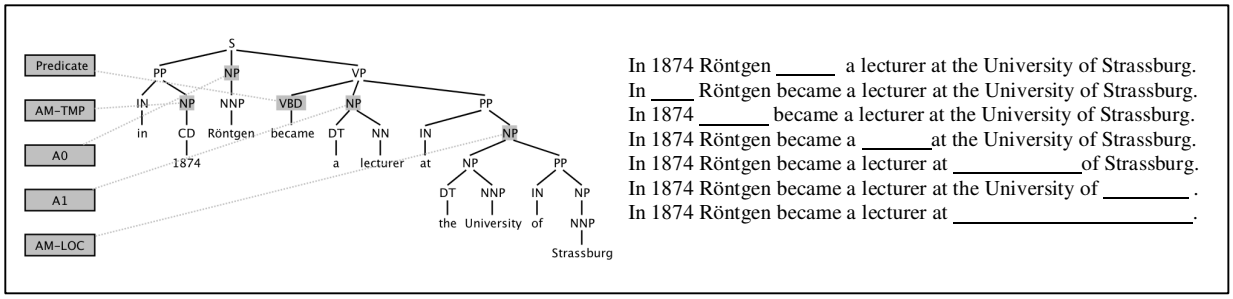

Figure 1 An example of the question generation process.

3.1 Sentence Selection

When learning about a new subject, a student will most likely want to learn about key concepts before moving onto more obscure details. As such, it is necessary to order target sentences in terms of their importance. This is fortunately very similar to the goals of automatic summarization, in which the selected sentences should be ordered by how central they are to the article.

As a result, we make use of our own implementation of SumBasic (Nenkova et al., 2006), a simple but competitive document summarization algorithm motivated by the assumption that sentences containing the article's most frequently occurring words are the most important. We thus use the SumBasic score for each sentence to order them as candidates for question construction.

3.2 Question Construction

We seek to empirically determine how to choose questions instead of relying on heuristics and rules for evaluating candidate surface forms. To do this, we cast question construction as a generate-and-filter problem: we overgenerate potential question/answer pairs from each sentence and train a discriminative classifier on human judgments of quality for those pairs. With the trained classifier, we can then apply this approach on unseen sentences to return the highest-scoring question/answer, all question/answer pairs scoring above a threshold, and so on.

Generation

Although it would be possible to select every word or phrase as a candidate gap, this tactic would produce a skewed dataset composed mostly of unusable questions, which would subsequently require much more annotation to discriminate good questions from bad ones. Instead we rely on syntactic and semantic constraints to reduce the number of questions that need annotation.

To generate questions we first run the source sentence through a constituency parser and a semantic role labeler (components of a state-of-the-art natural language toolkit from (Quirk et al., 2012)), with the rationale that important parts of the sentence will occur within a semantic role. Each verb predicate found within the roles then automatically becomes a candidate gap. From every argument to the predicate, we extract all child noun phrases (NP) and adjectival phrases (ADJP) as candidate gaps as well. Figure 1 illustrates this generation process.

Classification

To train the classifier for question quality, we aggregated per-question ratings into a single label (see Section 4 for details). Questions with an average rating of 0.67 or greater were considered as positive examples. This outcome was then paired with a vector of features (see Section 5) extracted from the source sentence and the generated question.

Because our goal is to score each candidate question in a meaningful way, we use a calibrated learner, namely L2-regularized logistic regression (Bishop 2006). This model's output is $p(class|features)$; in our case this is the posterior probability of a candidate receiving a positive label based on its features.

4 Corpus Construction

We downloaded 105 articles from Wikipedia's listing of vital articles/popular pages.[1] These articles represent a cross section of historical, social,

[1]http://en.wikipedia.org/wiki/Wikipedia:Vital_articles/Popular_pages

Source Sentence: *The large scale production of chemicals was an important development during the Industrial Revolution.*		
Question	Answer	Ratings
The _ _ _ _ _ _ _ _ _ _ _ _ of chemicals was an important development during the Industrial Revolution.	large scale production	◯Good ◉ Okay ◯Bad
The large scale production of _ _ _ _ _ _was an important development during the Industrial Revolution.	chemicals	◯Good ◉ Okay ◯Bad
The large scale production of chemicals was an important development during the _ _ _ _ _ _ _ _ _ _ _ _ .	Industrial Revolution	◉ Good ◯Okay ◯Bad

Figure 2: Example question rating HIT

and scientific topics. From each article we sampled 10 sentences using the sentence selection algorithm described in Section 3.1 for 50% of the sentences; the other 50% were chosen at random to prevent any possible overfitting to the selection method. These sentences were then processed using the candidate generation method from Section 3.2.

To collect training data outcomes for our question classifier, we used Amazon's Mechanical Turk (AMT) service to obtain human judgments of quality for each question. We initially considered asking about the quality of individual question/answer pairs in isolation, but in pilot studies we found poor agreement in this case; we noticed that the inability to compare with other possible questions actually made the task seem difficult and arbitrary. We thus instead presented raters with a source sentence and a list of up to ten candidate questions along with their corresponding answers (see Figure 2). Raters were asked to rate questions' quality as *Good*, *Okay*, or *Bad.* The task instructions defined a *Good* question as "one that tests key concepts from the sentence and would be reasonable to answer." An *Okay* question was defined as "one that tests key concepts but might be difficult to answer (the answer is too lengthy, the answer is ambiguous, etc.)." A *Bad* question was "one that asks about an unimportant aspect of the sentence or has an uninteresting answer that can be figured out from the context of the sentence." These ratings were binarized into a score of one for *Good* and zero for not-*Good* (*Okay* or *Bad*), as our goal was to find the probability of a question being truly *Good* (and not just *Okay*).[2]

[2] Heilman and Smith (2010a and b) asked raters to identify question deficiencies, including vague or obvious, but raters were not asked to differentiate between *Good* and *Okay*. Thus questions considered *Good* in their study would include *Okay*.

Thus far we have run 300 HITs with 4 judges per HIT. Each HIT consisted of up to 10 candidate questions generated from a single sentence. In total this yielded 2252 candidate questions with 4 ratings per question from 85 unique judges. We then wished to eliminate judges who were gaming the system or otherwise performing poorly on the task. It is common to do such filtering when using crowdsourced data by using the majority or median vote as the final judgment or to calibrate judges using expert judgments (Snow et al. 2008). Other approaches to annotator quality control include using EM-based algorithms for estimating annotator bias (Wiebe et al. 1999, Ipeirotis et al. 2010). In our case, we computed the distance for each judge from the median judgment (from all judges) on each question, then took the mean of this distance over all questions they rated. We removed judges with a mean distance two standard deviations above the mean distance, which eliminated the five judges who disagreed most with others.

In addition to filtering judges, we wanted to further constrain the data to those questions on which the human annotators had reasonable agreement, as it would not make sense to attempt to train a model to replicate judgments on which the annotators themselves could not agree. To do this, we computed the variance of the judgments for each question. By limiting the variance to 0.3, we kept questions on which up to 1 judge (out of 4) disagreed; this eliminated 431 questions and retained the 1821 with the highest agreement. Of these filtered questions, 700 were judged to be *Good* (38%).

To formally assess inter-rater reliability we computed Krippendorff's alpha (Krippendorff, 2004), a statistical measure of agreement applicable for situations with multiple raters and incomplete data (in our case not all raters provided ratings for all items). An alpha value of 1.0 indicates perfect agreement, and an alpha value of 0.0

indicates no agreement. Our original data yielded an alpha of 0.34, whereas after filtering judges and questions the alpha was 0.51. It should be noted that because Krippendorff's Alpha accounts for variability due to multiple raters and sample size, its values tend to be more pessimistic than many Kappa values commonly used to measure inter-rater reliability.

For others interested in working with this data, we have made our corpus of questions and ratings available for download at the following location: http://research.microsoft.com/~sumitb/questiongeneration.

5 Model Features

While intuition would suggest that selecting high-quality gaps for cloze questions should be a straightforward task, analysis of our features implies that identifying important knowledge depends on more complex interactions between syntax, semantics, and other constraints. In designing features, we focused on using commonly extracted NLP information to profile the answer (gap), the source sentence, and the relation between the two.

To enable extraction of these features, we used the MSR Statistical Parsing and Linguistic Analysis Toolkit (MSR SPLAT)[3], a state-of-the-art, web-based service for natural language processing (Quirk et al., 2012). Table 1 shows a breakdown of our feature categories and their relative proportion of the feature space. In the subsections below, we describe the intuitions behind our choice of features and highlight example features from each of these categories. An exhaustive list of the features can be found at the corpus URL listed in Section 4.

5.1 Token Count Features

A good question gives the user sufficient context to answer correctly without making the answer obvious. At the same time, gaps with too many words may be impossible to answer. Figure 3 shows the distributions of number of tokens in the answer (i.e., the gap) for *Good* and not-*Good* questions. As intuition would predict, the not-*Good* class has higher likelihoods for the longer answer lengths. In addition to the number and percentage of tokens in the answer features, we also included other token count features such as the number of tokens in the sentence, and the number of overlapping tokens between the answer and the remainder of the sentence.

Feature Category	Number of Features
Token Count	5
Lexical	11
Syntactic	112
Semantic	40
Named Entity	11
Wikipedia link	3
Total	182

Table 1: Breakdown of features by category

5.2 Lexical features

Although lexical features play an important role in system performance for several NLP tasks like parsing, and semantic role labeling, they require a large number of examples to provide practical benefit. Furthermore, because most sentences in Wikipedia articles feature numerous domain-specific terms and names, we cannot rely on lexical features to generalize across the variety of possible articles in our corpus. Instead we approximate lexicalization by computing densities of word categories found within the answer. The intuition behind these features is that an answer composed primarily of pronouns and stopwords will make for a bad question while an answer consisting of specific entities may make for a better question. Examples of our semi-lexical features include answer pronoun density, answer abbreviation density, answer capitalized word density, and answer stopword density.

5.3 Syntactic Features

The answer's structure relative to the sentence's structure provides information as to where better spans for the gap may exist. Similarly, part-of-speech (POS) tags give a topic-independent representation of the composition and makeup of the questions and answers. The collection of syntactic features includes the answer's depth with the sentence's constituent parse, the answer's location relative to head verb (before/after), the POS tag before the answer, the POS tag after the answer, and the answer bag-of-POS tags.

[3] http://research.microsoft.com/projects/msrsplat

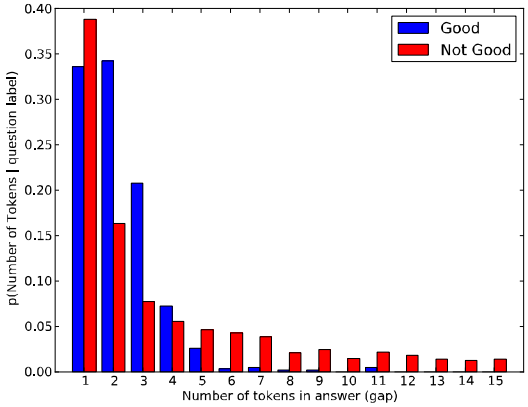

Figure 3: Distribution of number of tokens in the answer for *Good* and not-*Good* questions.

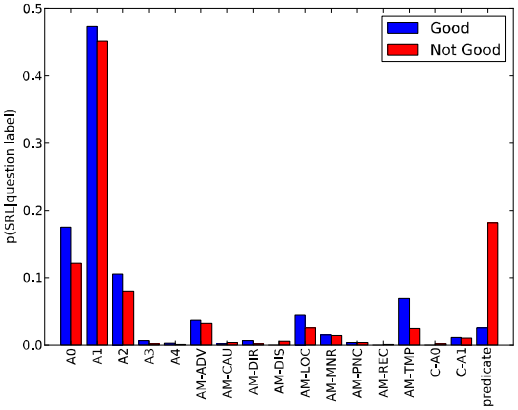

Figure 4: Distribution of semantic role labels for *Good* and not-*Good* questions.

5.4 Semantic Role Label Features

Beyond syntactic constraints, semantics can yield additional cues in identifying the important spans for questioning. Shallow-semantic parses like those found in Propbank (Palmer et al., 2005) provide a concise representation for linking predicates (verbs) to their arguments. Because these semantic role labels (SRLs) often correspond to the "who, what, where, and when" of a sentence, they naturally lend themselves for use as features for rating question quality. To compute SRL features, we used the MSR SPLAT's semantic role labeler to find the SRLs whose spans cover the question's answer, the SRLs whose spans are contained within the answer, and the answer's constituent parse depth within the closest covering SRL node.

To investigate whether judges keyed in on specific roles or modifiers when rating questions, we plotted the distribution of the answer-covering SRLs (Figure 4). This graph indicates that good answers are not associated with only a single label but are actually spread across all SRL classes. While the bulk of questions came from the arguments often corresponding to subjects and objects (ARG0-2, shown as A0-A2), we see that good and bad questions have mostly similar distributions over SRL classes. However, a notable exception are answers covered by verb predicates (shown as "predicate"), which were highly skewed with 190 of the 216 (88.0%) question/answer pairs exhibiting this feature labeled as *Bad*. Together these distributions may suggest that judges are more likely to rate gap-fill questions as Good if they correspond to questions of "who, what, where, and when" over questions pertaining to "why and how."

5.5 Named Entity Features

For many topics, especially in the social sciences, knowing the relevant people and places marks the first step toward comprehending new material. To reflect these concerns we use the named-entity tagger in the toolkit to identify the spans of text that refer to persons, locations, or organizations, which are then used to derive additional features for distinguishing between candidate questions. Example named-entity features include: answer named entity density, answer named entity type frequency (LOC, ORG, PER), and sentence named entity frequency.

Figure 5 shows the distribution of named entity types found within the answers for *Good* and not-*Good* questions. From this graph, we see that *Good* questions have a higher class-conditional probability of containing a named entity. Furthermore, we see that *Good* questions are not confined to a single named entity type, but are spread across all types. Together, these distributions indicate that while named entities can help to identify important gaps, the majority of questions labeled *Good* do not contain any named entity (515/700, i.e. 74%). This provides substantial evidence for generating questions for more than only named entities.

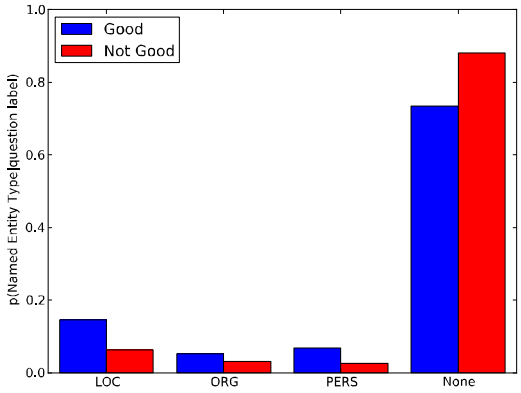

Figure 5: Distribution of answer named entity type for *Good* and not-*Good* questions.

5.6 Wikipedia Link Features

Wikipedia's markup language allows spans of text to link to other articles. This annotation inherently indicates a span of text as noteworthy, and can serve as evidence of an answer's importance. We use the presence of this markup to compute features such as answer link density, sentence link density, and the ratio of the number of linked words in the answer to the ratio of linked words in the sentence.

6 Model and Training

We chose logistic regression as our classifier because of its calibrated output of the class posterior; we combined it with an L2 regularizer to prevent overfitting. As the data likelihood is convex in the model parameters, we trained the model to maximize this quantity along with the regularization term using the L-BFGS algorithm for Quasi-Newton optimization (Nocedal, 1980). Evaluation was conducted with 10-fold cross validation, taking care to stratify folds so that all questions generated from the same source sentence are placed in the same fold. Results are shown in Section 7 below.

To ensure that we were not overly narrow in this model choice, we tested two other more powerful classifiers that do not have calibrated outputs, a linear SVM and a boosted mixture of decision trees (Caruana and Niculescu-Mizil, 2006); both produced accuracies within a percentage point of our model at the equal error rate.

7 Results and Discussion

Figure 6 shows ROC curves for our question quality classifier produced by sweeping the threshold on the output probability, using the raw collected data, our filtered version as described above, and a further filtered version keeping only those questions where judges agreed perfectly; the benefits of filtering can be seen in the improved performance. In this context, the true positive rate refers to the fraction of *Good* questions that were correctly identified, and the false positive rate refers to the fraction of not-*Good* questions that were incorrectly marked. At the equal error rate, the true positive rate was 0.83 and the false positive rate was 0.19**.**

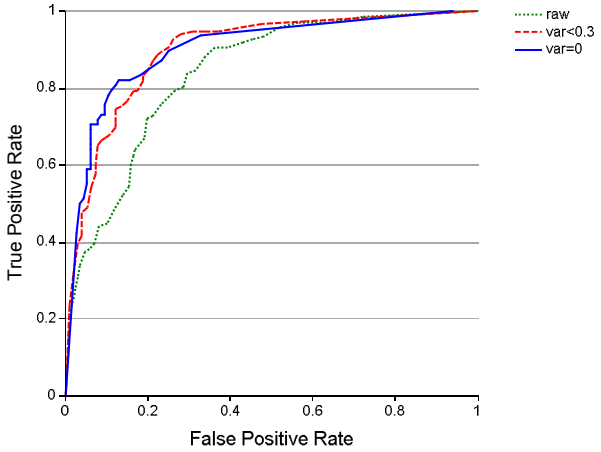

Figure 6: ROC for our model using unfiltered data (green dots), our filtered version (red dashes), and filtered for perfect agreement (blue line).

Choosing the appropriate operating point depends on the final application. By tuning the classifier's true positive and false positive rates, we can customize the system for several uses. For example, in a relatively structured scenario like compliance training, it may be better to reduce any possibility of confusion by eliminating false positives. On the other hand, a self-motivated learner attempting to explore a new topic may tolerate a higher false positive rate in exchange for a broader diversity of questions. The balance is subtle, though, as ill-formed and irrelevant questions could leave the learner bored or frustrated, but alternatively, overly conservative question classification could potentially eliminate all but the most trivial questions.

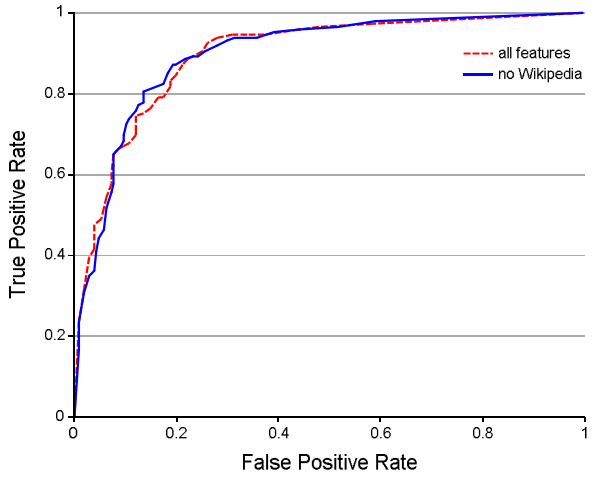

Figure 7: ROC for our model with (red dash) and without (blue line) Wikipedia-specific features.

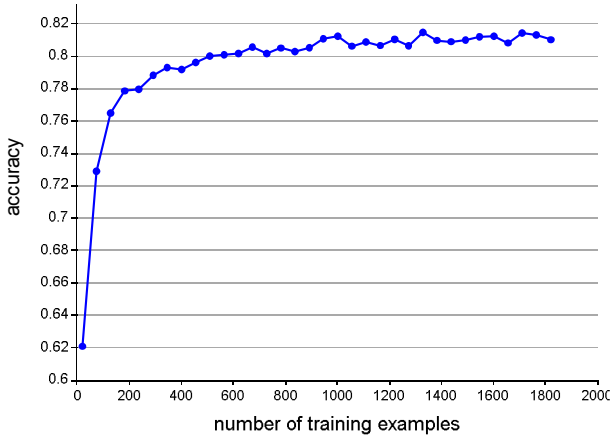

Figure 8: Classifier learning curve; each point represents mean accuracy over 40 folds.

We next wanted to get a sense of how well the model would generalize to other text, and as such ran an analysis of training the classifier without the benefit of the Wikipedia-specific features (Figure 7). The resulting model performs about the same as the original on average over the ROC, slightly better in some places and slightly worse in others. We hypothesize the effect is small because these features relate only to Wikipedia entities, and the other named entity features likely make them redundant.

Finally, to understand the sensitivity of our model to the amount of training data, we plot a learning curve of the question classifier's accuracy by training it against fractions of the available data (Figure 8). While the curve starts to level out around 1200 data points, the accuracy is still rising slightly, which suggests the system could achieve some small benefits in accuracy from more data.

7.1 Error Analysis

To explore the nature of our system's misclassifications we examine the errors that occur at the equal error rate operating point. For our system, false positive errors occur when the system labels a question as *Good* when the raters considered it not-*Good*. Table 2 lists three examples of this type of error. The incorrect high score in example 1 ("Greeks declared ___") suggests that system performance can be improved via language modeling, as such features would penalize questions with answers that could be predicted mostly by word transition probabilities. Similarly, when classifying questions like example 2 ("such as ____ for a mathematical function"), the system could benefit from some measure of word frequency or answer novelty. While our model included a feature for the number of overlapping words between the question and the answer, the high classifier score for example 3 ("atop _____, the volcano"), suggests that this can be solved by explicitly filtering out such questions at generation time.

With false negative errors the judges rated the question as *Good*, whereas the system classified it as *Bad*. The question and answer pairs listed in Table 3 demonstrate some of these errors. In example 1 ("where Pompey was soon ___"), the system was likely incorrect because a majority of questions with verb-predicate answers had *Bad* ratings (only 12% are *Good*). Conversely, classification of example 2 ("Over the course of decades…") could be improved with a feature indicating the novelty of the words in the answers. Example 3 ("About 7.5% of the...") appears to come from rater error or rater confusion as the question does little to test the understanding of the material.

While the raters considered the answer to example 4 as *Good*, the low classifier score argues for different handling of answers derived from long coordinated phrases. One alternative approach would be to generate questions that use multiple gaps. Conversely, one may argue that a learner may be better off answering any one of the noun phrases like *palm oil* or *cocoa* in isolation.

	Question	Answer	Confidence
1	*In 1821 the Greeks declared ___ on the sultan.*	*war*	*0.732*
2	*He also introduced much of the modern mathematical terminology and notation, particularly for mathematical analysis, such as _____ of a mathematical function.*	*the notion*	*0.527*
3	*Not only is there much ice atop _____, the volcano is also being weakened by hydrothermal activity.*	*the volcano*	*0.790*

Table 2: Example false positives (human judges rated these as not-*Good)*

	Question	Answer	Confidence
1	*Caesar then pursued Pompey to Egypt, where Pompey was soon ____.*	*murdered*	0.471
2	*Over the course of decades, individual wells draw down local temperatures and water levels until _____ is reached with natural flows.*	*a new equilibrium*	0.306
3	*About 7.5% of world sea trade is carried via the canal ____.*	*today*	0.119
4	*Asante and Dahomey concentrated on the development of "legitimate commerce" in _____, forming the bedrock of West Africa's modern export trade,*	*the form of palm oil, cocoa, timber, and gold.*	0.029

Table 3: Example false negatives (human judges rated these *Good*)

7.2 Feature Analysis

To ensure that all of the gain of the classifier was not coming from only a handful of isolated features, we examined the mean values for each feature's learned weight in the model over the course of 10 cross-validation folds, and then sorted the means for greater clarity (Figure 8). The weights indeed seem to be well distributed across many features.

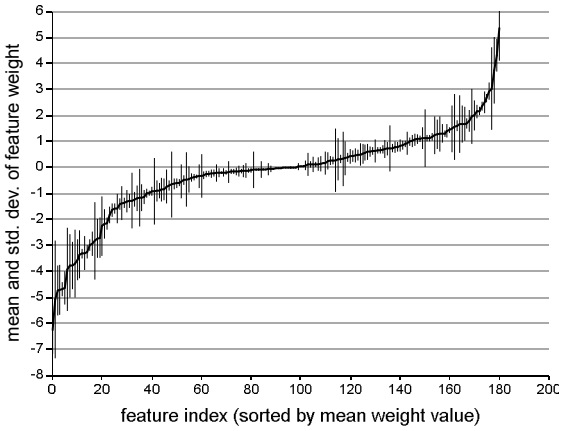

Figure 8: Feature weight means and standard deviations.

8 Discussion and Future Work

We have presented a method that determines which gaps in a sentence to ask questions about by training a classifier that largely agrees with human judgments on question quality. We feel this effort is complementary to the past work on question generation, and represents another step towards helping self-motivated learners with automatically generated tests.

In our future work, we hope to expand the set of features as described in Section 7. We additionally intend to cast the sentence selection problem as a separate learning problem that can also be trained from human judgments.

References

Manish Agarwal and Prashanth Mannem. 2011. Automatic Gap-fill Question Generation from Text Books. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications.* Portland, OR, USA. pages 56-64.

Richard C. Anderson and W. Barry Biddle. 1975. On asking people questions about what they are reading. In G. Bower (Ed.) *Psychology of Learning and Motivation,* 9:90-132.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning.* New York: Springer, 2006.

Jonathan C. Brown, Gwen A. Frishkoff, and Maxine Eskenazi. 2005. Automatic Question Generation for Vocabulary Assessment. In *Proceedings of HLT/EMNLP 2005.* Vancouver, Canada: Association for Computational Linguistics. pages 819-826.

Rich Caruana and Alexandru Niculescu-Mizil. 2006. An Empirical Comparison of Supervised Learning Algorithms. In *Proceedings of ICML 2006.*

Chris Callison-Burch and Mark Dredze. 2010. Creating Speech and Language Data with Amazon's Mechanical Turk. In *Proceedings of NAACL 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Los Angeles, CA. pages 1-12.

Wei Chen, Gregory Aist, and Jack Mostow. 2009. Generating Questions Automatically from Informational Text. In S. Craig & S. Dicheva (Ed.), *Proceedings of the 2nd Workshop on Question Generation.*

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. 2011. Exploring linguistically-rich patterns for question generation. In *Proceedings of the UCNLG + eval: Language Generation and Evaluation Workshop*. Edinburgh, Scotland: Association for Computational Linguistics. Pages 33-38.

Michael Heilman and Noah A. Smith. 2010a. Good Question! Statistical Ranking for Question Generation. In *Proceedings of NAACL/HLT 2010.* pages 609-617.

Michael Heilman and Noah A. Smith. 2010b. Rating Computer-Generated Questions with Mechanical Turk. In *Proceedings of NAACL 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Los Angeles, CA. pages 35-40.

Ayako Hoshino and Hiroshi Nakagawa. 2005. A real-time multiple-choice question generation for language testing - a preliminary study -. In *Proceedings of the 2nd Workshop on Building Educational Applications Using NLP*. Ann Arbor, MI, USA: Association for Computational Linguistics. pages 17-20.

Panagiotis G. Ipeirotis, Foster Provost, Jing Wang . 2010. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP'10).*

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology*. Thousand Oaks, CA: Sage.

Ruslan Mitkov and Le An Ha. 2003. Computer-Aided Generation of Multiple-Choice Tests. *Proceedings of the HLT-NAACL 2003 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17-22.

Ruslan Mitkov, Le An Ha, and Nikiforos Karamanis. 2006. A computer-aided environment for generating multiple choice test items. *Natural Language Engineering,* 12(2): 177-194.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A Compositional Context Sensitive Multidocument Summarizer. In *Proceedings of SIGIR 2006.* pages 573-580.

Rodney D. Nielsen. 2008. Question Generation: Proposed Challenge Tasks and Their Evaluation. In V. Rus, & A. Graesser (Ed.), In *Proceedings of the Workshop on the Question Generation Shared Task and Evaluation Challenge.* Arlington, VA.

Jorge Nocedal. 1980. Updating Quasi-Newton Matrices with Limited Storage. *Mathematics of Computation,* 35:773-782.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, vol. 31, no. 1, pp. 71--106.

Chris Quirk, Pallavi Choudhury, Jianfeng Gao, Hisami Suzuki, Kristina Toutanova, Michael Gamon, Wen-tau Yih, and Lucy Vanderwende. 2012. MSR SPLAT, a language analysis toolkit. In *Proceedings of NAACL HLT 2012 Demonstration Session*. http://research.microsoft.com/projects/msrsplat .

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev and Cristian Moldovan. 2010. Overview of The First Question Generation Shared Task Evaluation Challenge. In *Proceedings of the Third Workshop on Question Generation.* Pittsburgh, PA, USA. pages 45-57.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and Fast—but is it Good? Evaluating non-Expert Annotations for Natural Language Tasks. In *Proceedings of EMNLP'08*. pages 254-263.

Lucy Vanderwende. 2008. The Importance of Being Important: Question Generation. In *Proceedings of the 1st Workshop on the Question Generation Shared Task Evaluation Challenge, Arlington, VA.*

Janyce M. Wiebe, Rebecca F. Bruce and Thomas P. O'Hara. 1999. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of ACL 1999.*

John H. Wolfe. 1976. Automatic question generation from text - an aid to independent study. In *Proceedings of the ACM SIGCSE-SIGCUE technical symposium on Computer science and education.* New York, NY, USA: ACM. pages 104-112.