Judging the Quality of Automatically Generated Gap-fill Question using Active Learning

Nobal B. Niraula and Vasile Rus
Department of Computer Science and Institute for Intelligent Systems
The University of Memphis
Memphis, TN, 38152, USA
{nbnraula,vrus}@memphis.edu

Abstract

In this paper, we propose to use active learning for training classifiers to judge the quality of gap-fill questions. Gap-fill questions are widely used for assessments in education contexts because they can be graded automatically while offering reliable assessment of learners' knowledge level if appropriately calibrated. Active learning is a machine learning framework which is typically used when unlabeled data is abundant but manual annotation is slow and expensive. This is the case in many Natural Language Processing tasks, including automated question generation, which is our focus. A key task in automated question generation is judging the quality of the generated questions. Classifiers can be built to address this task which typically are trained on human labeled data. Our evaluation results suggest that the use of active learning leads to accurate classifiers for judging the quality of gap-fill questions while keeping the annotation costs in check. We are not aware of any previous effort that uses active learning for question evaluation.

1 Introduction

Recent explosion of massive open online courses (MOOCs) such as Coursera[1] and Udacity[2] and the success of Intelligent Tutoring Systems (ITSs), e.g. AutoTutor (Graesser et al., 2004) and DeepTutor (Rus et al., 2013), at inducing learning gains comparable to human tutors indicate great opportunities for online education platforms. These systems typically deliver knowledge to learners via video streaming or direct interaction with the system, e.g. dialogue based interaction. If adaptive to individual learners, such online platforms for learning must assess learners' knowledge before, during, and after students' interaction with the platform. For instance, in order to identify knowledge deficits before and/or after a session a pre- and/or post-test can be used. The knowledge deficits discovered based on the pre-test can guide the online platform to select appropriate instructional tasks for the learner. Furthermore, the pre- and post-test can be used to measure the learning gains with the online platform, e.g. by subtractic the pre-test score from the post-test score. The bottom line is that assessment is critical for adaptive instruction. Various kinds of questions are used to assess students' knowledge levels varying from True/False questions to multiple choice questions to open answer questions.

Indeed, a main challenge in online learning platforms such as MOOCs and ITSs is test construction (assessment question generation). Automated test construction is a demanding task requiring significant resources. Any level of automation in question generation would therefore be very useful for this expensive and time-consuming process. In fact, it has been proven that computer-assisted test construction can dramatically reduce costs associated with test construction activities (Pollock et al., 2000). Besides test construction, automatic question generation are very useful in several other applications such as reading comprehension (Eason et al., 2012), vocabulary assessment (Brown et

[1] http://www.coursera.org
[2] http://www.udacity.com

Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications, 2015, pages 196–206,
Denver, Colorado, June 4, 2015.

al., 2005), and academic writing(Liu et al., 2012). Consequently, particular attention has been paid by Natural Language Processing (NLP) and educational researchers to automatically generating several types of questions. Some examples include *multiple choice questions* (Mitkov et al., 2006; Niraula et al., 2014), *gap-fill* questions (Becker et al., 2012) and *free-response* questions (Mazidi and Nielsen, 2014a; Heilman and Smith, 2009). The more general problem of question generation has been systematically addressed via shared tasks (Rus et al., 2010).

Mitkov et al. (2006) reported that automatic question construction followed by manual correction is more time-efficient than manual construction of the questions alone. Automated method for judging the question quality would therefore make the question generation process much more efficient. To this end, we present in this paper an efficient method to rank *gap-fill* questions, a key step in generating the questions. We formulate the problem next.

1.1 Gap-fill Question Generation

Gap-fill questions are *fill-in-the-blank* questions consisting of a sentence/paragraph with one or more gaps (blanks). A typical gap-fill question is presented below:

Newton's _____ law is relevant after the mover doubles his force as we just established that there is a non-zero net force acting on the desk then.

The gap-fill question presented above has a word missing (i.e. a gap). A gap-fill question can have one more than one gaps too. Students (test takers) are supposed to predict the missing word(s) in their answer(s). Gap-fill questions can be of two types: with alternative options (*key and distractors*) and without choices. The former are called *cloze* questions and the latter are called *open-cloze* questions. In this paper, we use the term gap-fill question as an alternative to open-cloze question.

The attractiveness of gap-fill questions is that they are well-suited for automatic grading because the correct answer is simply the original word/phrase corresponding to the gap in the original sentence. As a result they are frequently used in educational contexts such as ITSs and MOOCs.

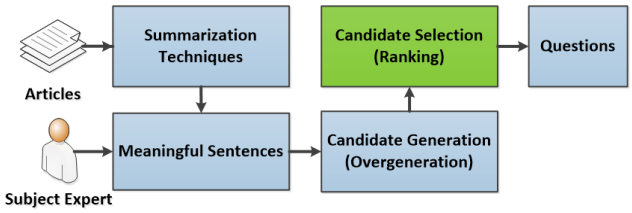

Figure 1: A pipeline for gap-fill question generation

A typical pipeline to automatically generate gap-fill questions is shown in Figure 1. It follows the three steps paradigm for question generation (Rus and Graesser, 2009) : *Sentence Selection*, *Candidate Generation* (overgeneration) and *Candidate Selection* (ranking).

Step 1 - Sentence Selection: To generate gap-fill questions, a set of meaningful sentences are needed first. The sentences can be selected from a larger source, e.g. a chapter in a textbook, using particular instructional criteria such as being difficult to comprehend or more general informationl criteria such as being a good summary of the source (Mihalcea, 2004) or directly from subject matter experts.

Step 2 - Candidate Generation: This step generates a list of candidate questions (*overgeneration*) from the target sentences selected in Step 1. The simplest method might be a brute force approach which generates candidate questions by considering each word (or a phrase) as a gap. A more advanced technique may target the content words as gaps or exploit the arguments of semantic roles for the gaps (Becker et al., 2012). An example of overgeneration of questions is shown in Table 1.

Step 3 - Candidate selection: Not all of the questions generated in the candidate generation step are of the same quality. The classes can be *Good*, *Okay* and *Bad* as in Becker *et al.* (2012) or simply the binary classes *Good* and *Bad*. *Good* questions are the questions that ask about key concepts from the sentence and are reasonable to answer, *Okay* questions are questions that target the key concepts but are difficult to answer (e.g. too long, ambiguous), and *Bad* questions are questions which ask about unimportant aspect of the sentence or their answers are easy to guess from the context. The candidate selection step is about rating the question candidates. Supervised machine learning models are typically em-

Bad	..... net force is equal to the mass times its acceleration.
Good	The ..... force is equal to the mass times its acceleration.
Good	The net ..... is equal to the mass times its acceleration.
Good	..... is equal to the mass times its acceleration.
Bad	The net force ..... equal to the mass times its acceleration.
Okay	The net force is ..... to the mass times its acceleration.
Bad	The net force is equal ..... the mass times its acceleration.
Good	The net force is equal to the ..... times its acceleration.
Okay	The net force is equal to the mass ..... its acceleration.
Bad	The net force is equal to the mass times ..... acceleration.

Table 1: *Typical overgenerated questions from a sentence with their ratings **Good**, **Okay** and **Bad**.*

ployed in the form of classifiers to label the candidate questions as Good, Okay, or Bad.

1.2 Question Quality

Question quality can be judged linguistically or pedagogically. In linguistic evaluation, questions are evaluated with respect to whether they are grammatically and semantically correct. In pedagogical evaluation, questions are evaluated to see whether they are helpful for understanding and learning the target concepts. Our focus here is on the pedagogical evaluation of automatically generated gap-fill questions since they are always linguistically correct.

The third step i.e. candidate selection is expensive when supervised approaches are used because model building in supervised learning requires large amount of human annotated examples. The advantage of supervised methods, however, is that their performances are in general better than, for instance, that of unsupervised methods. As such, ideally, we would like to keep the advantages of supervised methods while reducing the costs of annotating data. Such a method that offers a good compromise between annotation costs and performance is *active learning*, which we adopt in this work. Such models are always attractive choices especially when there is a limited budget e.g. fixed annotation time / cost, a highly probable case.

Active learning and *interactive learning* are two well-known approaches that maximize performance of machine learning methods for a given budget. They are successfully applied for rapidly scaling dialog systems (Williams et al., 2015), parts-of-speech tagging (Ringger et al., 2007), sequence labeling (Settles and Craven, 2008), word sense disambiguation (Chen et al., 2006), named entity tagging (Shen et al., 2004), etc. Instead of selecting and presenting to an annotator a random sample of unlabeled instances to annotate, these approaches intelligently rank the set of unlabeled instances using certain criteria (see Section 3) and present to the annotator the best candidate(s). This characteristic of active learning and interactive labeling hopefully demands fewer instances than random sampling to obtain the same level of performance.

In this paper, we propose an active learning based approach to judge the quality of gap-fill questions with the goal of reducing the annotation costs. We are not aware of any previous effort that uses active learning for question generation. We chose active learning particularly because it is well-suited when unlabeled data is abundant but manual annotation is tedious and expensive. As mentioned, this is the case in gap-fill question question generation in overgeneration approaches when plenty of questions are available but their quality needs to be specified. The remaining challenge is to judge the quality of these questions. Our plan is to build a probabilistic classifier at reduced costs that would automatically label each candidate questions as *good* or *bad* using an active learnign approach.

The rest of the paper is organized as follows. In Section 2, we present the relevant works. In Section 3, we present the active learning techniques that we are going to employ. In Section 4 and Section 5, we describe our experiments and results respectively. We present the conclusions in Section 6.

2 Related Works

Currently, statistical and machine learning based approaches are the most popular approaches that are used to rank the automatically generated questions of various kinds e.g. free-response (e.g. What, When etc.) and gap-fill questions. For example, Heilman et al. (2010) used logistic regression, a supervised method, to predict the acceptability of each free-response question candidate. The candidate questions were automatically generated by using a set of rules. They used fifteen native English-speaking university students for the construction of training examples required for building the logistic regression model.

Hoshino and Nakagawa (2005) proposed a machine learning approach to generate multiple-choice questions for language testing. They formed a question sentence by deciding the position of the gap i.e. missing word(s). To decide whether a given word can be left blank (i.e. serve as a gap) in the declarative stem, they trained classifiers using the training instances which were generated by collecting fill-in-the-blank questions from a TOEIC preparation book. The positive examples were the exact blank positions in the question from the book whereas the negative examples were generated by shifting the blank position.

Similarly, Becker *et al.* (2012) proposed *Mind the Gap* system that applied logistic regression to rank automatically generated gap-fill questions. They used text summarization technique to select useful sentences from text articles for which gap-fill questions are to be generated. From each of the selected sentence, it generated potential gap-fill candidates using semantic constraints. Each candidate question was then labeled by four Amazon's Mechanical Turkers to one of *Good*, *Bad* and *Okay* classes. In total, 85 unique Turkers were involved in the annotation. The data set was used to build a logistic regression classifier and ranked the candidate questions. They reported that the classifier largely agreed with the human judgment on question quality.

In recent works Mazidi and Nielsen (2014a; 2014b) generated free-response questions from sentences by using the patterns which were manually authored by exploiting the semantic role labels. They evaluated the questions linguistically and pedagogically using human annotators and reported that their systems produced higher quality questions than comparable systems. The main limitation of their approaches is that they do not exploit the examples obtained from the annotation process to evaluate unseen (or not yet evaluated) questions. Moreover, their approaches do not provide any ranking for the questions they generated using those patterns.

3 Active Learning for Judging Question Quality

As mentioned before, active learning fits well when abundant data can be available but manual labeling costs are high. As a result, the technique has been applied to many NLP tasks such as text classification, Word Sense Disambiguation, sequence labeling, and parsing. We use active learning for guiding our annotation process for judging the quality of automatically generated gap-fill questions.

3.1 Active Learning Algorithms

An active learning system mainly consists of a classification model and querying algorithm. Typically the classification models are the probabilistic classifiers such as Naïve Bayes and Logistic Regression which provide a class probability distribution for a given instance. Querying algorithms/functions actively choose unlabeled instance samples by exploiting these probabilities.

We follow the standard pool-based active learning algorithm as shown in Algorithm 1. It starts with a set of initially labeled instances (seed examples) and a set of unlabeled instances (U). A new model is built using the labeled examples in L. Next, a batch of instances are extracted from the unlabeled set U using a query function $f(.)$ and then the selected instances are labeled by human judges. The new labeled instances are added to the labeled list L. The process repeats until a stopping criterion is met. The criteria could be the number of examples labeled, expected accuracy of the model, or something else.

3.1.1 Querying Algorithms

Many query functions exist. They differ on how they utilize the class probability distributions. We use two variants of query functions: uncertainty sampling and query by committee sampling.

Input: Labeled instances L, unlabeled instances U, query batch size B, query function $f(.)$;
while *some stopping criterion* **do**
  θ = Train the model using L;
  for i = *1 to* B **do**
    $b_i^* = \arg\max_{u \in U} f(u)$;
    $L = L \cup < b_i^*, label(b_i^*) >$;
    $U = U - b_i^*$;
  end
end

Algorithm 1: Pool-based active learning algorithm

A. Query By Uncertainty or Uncertainty Sampling

Uncertainty sampling chooses the samples for which the model's predictions are least certain. These examples reside very near to the decision boundary. We use three functions that predict the samples in the decision boundary.

(a) Least Confidence: This function chooses the sample x that has the highest $f_{LC}(.)$ score and is defined as : $f_{LC}(x) = 1 - P(y^*|x;\theta)$ where y^* is the most likely class predicted by the model (Settles and Craven, 2008).

(b) Minimum Margin: This function chooses the sample x that has the least $f_{MM}(.)$ score and is defined as: $f_{MM}(x) = |P(y_1^*|x;\theta) - P(y_2^*|x;\theta)|$ where y_1^* and y_2^* are the first and the second most likely classes predicted by the model (Chen et al., 2006).

(c) Entropy: This function chooses the sample x that has the highest entropy i.e. $f_{EN}(.)$ score and is defined as: $f_{EN}(x) = -\sum_{c=1}^{C} P(y_c|x;\theta) * \log(P(y_c|x;\theta))$ where C is the total number of classes (Chen et al., 2006).

B. Query By Committee

Our query by committee sampling algorithm consists of a committee of models. These models are trained on the same labeled examples but learn different hypotheses. We compute for a given instance the class distribution mean over all committee members and assume that the mean scores represent the votes received from the committee. Next we apply $f_{LC}(.)$, $f_{MM}(.)$ and $f_{EN}(.)$ over the mean class distribution and view them as selection scores.

4 Experiments

In this section we describe our experiments in detail.

4.1 Data set

Although an active learning system doesn't require all the unannotated instances to be labeled initially, having such an annotated data set is very useful for simulations since it allows us to conduct experiments to inverstigate active learning, in our case, for judging the quality of automatically generated questions. To this end, we used the existing data set called *Mind the Gap data set* which was created and made publicly available by Becker *et al.* (2012) [3]. The data set consists of 2,252 questions generated using sentences extracted from 105 Wikipedia's articles across historical, social, and scientific topics. Each question was rated by four Amazon Mechanical Turkers as *Good*, *Okay*, or *Bad* (see definitions in Section 1.1).

For our experiments, we binarized the questions into *positive* and *negative* examples. We considered a question *positive* when all of its ratings were *Good* or at most one rating was *Okay* or *Bad*. The rest of the questions were considered as *negative* examples. This way we obtained 747 positive and 1,505 were negative examples. The chosen requirement for being a positive example was needed in order to focus on high quality questions.

4.2 Features

In order to build models to judge the quality of questions, we implemented five types of features as in Becker et al. (2012) including *Token Count, Lexical, Syntatic, Semantic* and *Named Entity*. In total we had 174 features which are summarized in Table 2. The numbers inside parentheses are the indices of the features used.

Questions with many gaps (with many missing words) are harder to answer. Similarly, gaps with many overlapped words with the remaining words in the question are not suitable since they can be easily inferred from the context. We used 5 different *Token Count* features to capture such properties. We also used 9 *Lexical* features to capture different

[3] http://research.microsoft.com/šumitb/questiongeneration

Type	Features
Token Count - 5	no. of tokens in answer(1) and in sentence(2), % of tokens in answer (3), no.(4) and %(5) of tokens in answer matching with non-answer tokens
Lexical - 9	% of tokens in answer that are capitalized words(6), pronouns(7), stopwords(8), and quantifiers(9), % of capitalized words(10) and pronouns(11) in sentence that are in answer, does sentence start with discourse connectives ?(12), does answer start with quantifier ?(13), does answer end with quantifier ?(14)
Syntatic - 116	is answer before head verb ? (15), depth of answer span in constituent parse tree (16), presence/absence of POS tags right before the answer span(17-54), presence/absence of POS tags right after the answer span(55-92), no. of tokens with each POS tag in answers(93-130)
Semantic - 34	Answer covered by (131-147), answer contains(148-164) the semantic roles: {A0, A1, A2, A3, A4, AM-ADV, AM-CAU, AM-DIR, AM-DIS, AM-LOC, AM-MNR, AM-PNC, AM-REC, AM-TMP, CA0, CA1, Predicate}
Named Entities - 11	does answer contain a LOC(165), PERS(166), and ORG(167) named entities ? does non-answer span contain a LOC(168), PERS(169), and ORG(164) named entities ? no. (170) and % (171) of tokens in answer that are named entities, no. (172) and % (173) of tokens in sentence that are named entities, % of named entities in sentence present in answer (174)

Table 2: List of features used

statistics of pronouns, stop words, quantifiers, capitalized words, and discourse connectives. Similarly, we used 116 *Syntatic* features that include mostly binary features indicating presence/absence of a particular POS tag just before the gap and just after the gap, and number of occurrences of each POS tag inside the gap. Our semantic features includes 34 binary features indicating whether the answer contained a list of semantic roles and whether semantic roles cover the answer. In addition, we used 11 *Named Entities* features to capture presence/absence of LOC, PERS and ORG entities inside the answer and outside the answer. We also computed the entity density i.e. number of named entities present in the answer. We used Senna tool for getting semantic roles (Collobert et al., 2011) and Stanford CoreNLP package (Manning et al., 2014) for getting POS tags and named entities.

5 Results and Discussions

We conducted a number of experiments to see how active learning performs at judging the quality of questions at different settings: type of classifiers (simple and committee), evaluation metrics (accuracy and F-Measure), seed data size, batch size, and sampling algorithms. An experiment consists of a number of runs. In each run, we divided the data set into three folds using stratified sampling. We considered one of the folds as the *test data set* and

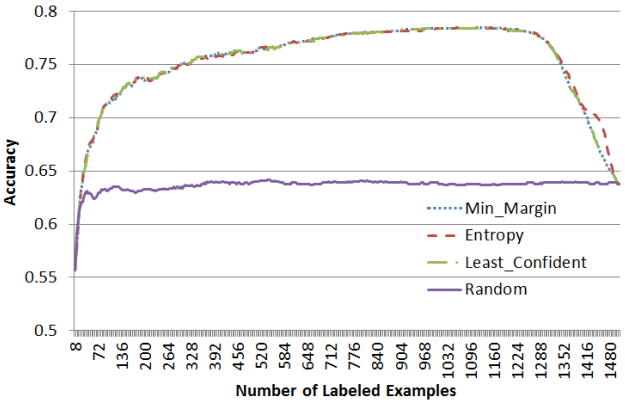

Figure 2: Full Simulation for Naïve Bayes Accuracy

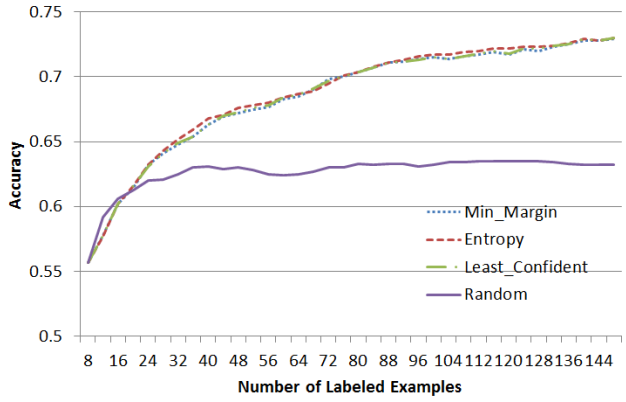

Figure 3: Close-up view of Naïve Bayes Accuracy

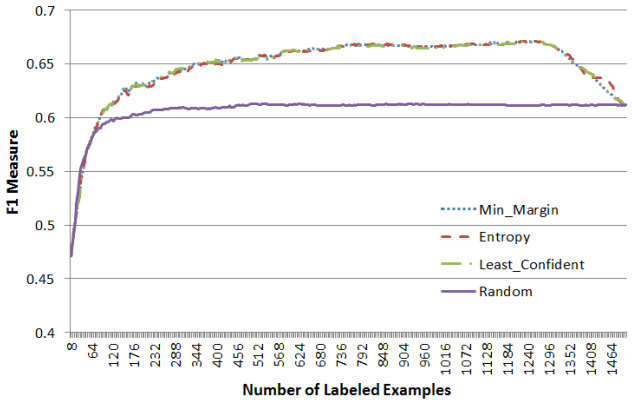

Figure 4: Full Simulation for Naïve Bayes F1

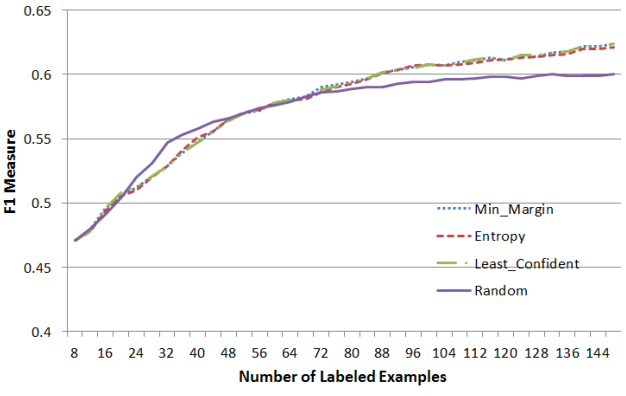

Figure 5: Close-up view of Naïve Bayes F1

merged the other two to construct the *unlabeled data set* (U). Remember that our data set is already labeled but we pretended that it is unlabeled U. Typically, the selected instances from U have to be labeled by a human. Since we already know all the labels in the data set, we mimic the human labeling by simply using the existing labels. This allows us to conduct several experiments very efficiently.

In the first experiment, we compared the various sampling techniques in terms of their impact of the overall performance of question quality classifier. To this end, we randomly selected 8 examples (four positive and 4 negative) from U for the seed data set, removed them from U and put them into the labeled data set (L). We then built a Naïve Bayes model for judging the quality of questions using L. All the machine learning algorithms we used are available in Weka (Hall et al., 2009). Next, we applied a given sampling strategy to select 4 best examples (i.e. a batch of size 4) to be labeled. These new labeled examples were added to L and the question quality classifier was retrained with this extended data set. We used the test data subset to evaluate the question quality classifier at each iteration and report accuracy and F-measure. The process was iterated until the unlabeled data set U was empty.

We used the four sampling algorithms (i.e. least confidence, minimum margin, entropy and random) and report results in terms of average across 100 different runs; in each such run we ran the active learning approach entirely on all the data we had available. Figure 2 and Figure 4 present the accuracy and F1 scores of Naïve Bayes for each of the sampling algorithms with respect to the number of labeled instances used. Figure 3 and Figure 5 are close-ups of leftmost part of the curves in Figure 2 and Figure 4, respectively. As we can see, all uncertainty sampling methods (Min-margin, Entropy and Least confident) outperformed random sampling for both accuracy and F1 measures after few annotations were made. For instance, with 200 examples selected by active learning, the model provided 10% more in accuracy and 4% more in F1 measure compared to the case when the same number of instances were used by sample randomly. It is a promising observation that can save annotation budgets significantly. Moreover, close-up graphs show that all three uncertainty sampling approaches rival each other. Note that all the sampling methods converged (i.e. have same accuracy and F1 measure) at the end of the simulation. It is normal because they would have the same set of labeled instances by then.

In the second experiment, we formed a committee of three probabilistic classifiers provided by Weka:

Naïve Bayes, Logistic Regression, and SMO. These classifiers learnt different hypotheses from the same set of training examples. As discussed in Section 3.1.1, we generated three models from the same labeled set of examples and computed mean probability distributions. For this experiment, we set seed size of 8, batch size of 4, and 100 runs as in experiment 1 and measured the performances of the sampling algorithms. Figure 6 and Figure 8 show the accuracy and F-measure for several sampling strategies as a function of the number of annotated examples. Figure 7 and Figure 9 are the close-up views for Figure 6 and Figure 8 respectively. Again, the uncertainty based sampling algorithms are very competitive to each other and they outperform random sampling significantly in both accuracy and F-measure. This suggests that committee based active learning is also useful for checking question quality.

To get an idea of the level of annotation savings when using active learning, consider we have a budget for annotating about 160 instances. With this budget (in Figure 6), uncertainty sampling algorithms provide 70% accuracy whereas random sampling provides only 65% accuracy. To attain 70% accuracy, random sampling needs at least 360 samples (i.e. 200 examples more) to be labeled. With 360 samples, uncertainty sampling algorithms provide 74% accuracy. Similar observations can be made when focusing on the F-measure. These observations clearly show the effectiveness of using active learning for judging the quality of automatically generated questions.

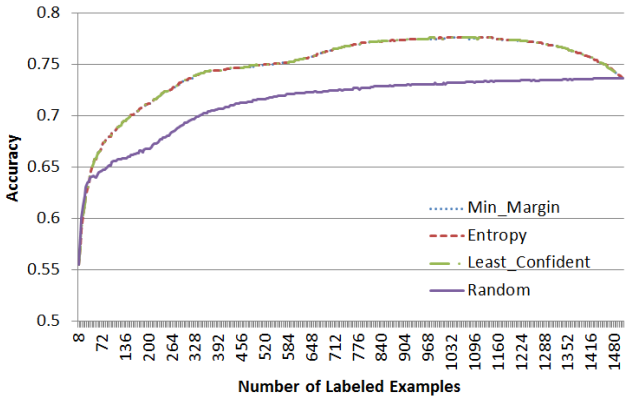

Figure 6: Full Simulation for Committee Accuracy

In a third experiment, we focused on the effect of

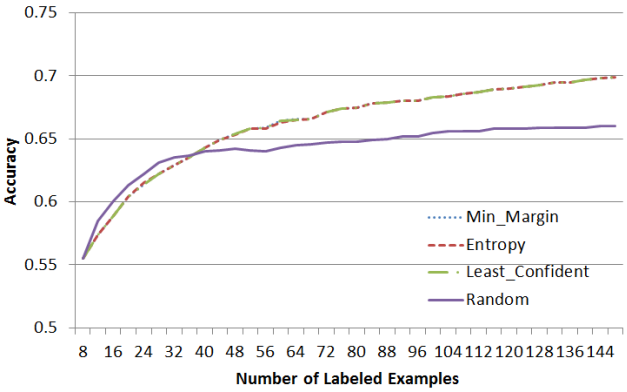

Figure 7: Close-up view of Committee Accuracy

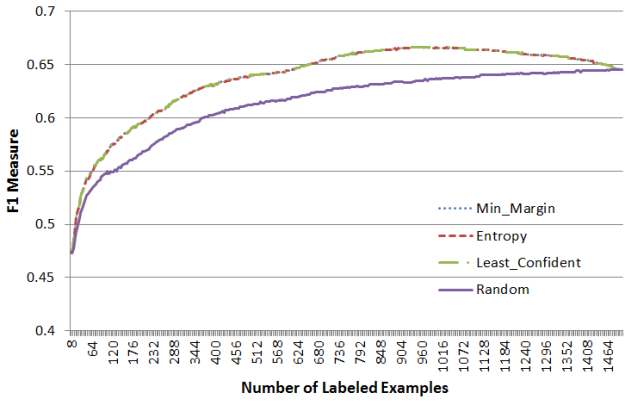

Figure 8: Full Simulation for Committee F1

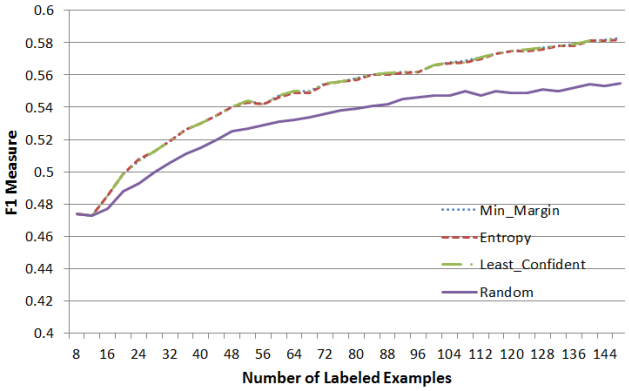

Figure 9: Close-up view of Committee F1

the batch size on the behavior of the active learning approach. Note that we generate a new model as soon as a new batch of labeled instances is ready. For instance, a batch size of 2 means as soon as the annotators provide two annotated instances, we add them to the labeled set and generate a new model

from all the available labeled instances. The new model is generally a better one as it is trained on a larger training set than the previous one. However, the smaller the batch size the larger the computational cost because we need to generate a model frequently. So, a balance between the computation cost and the better model should be determined.

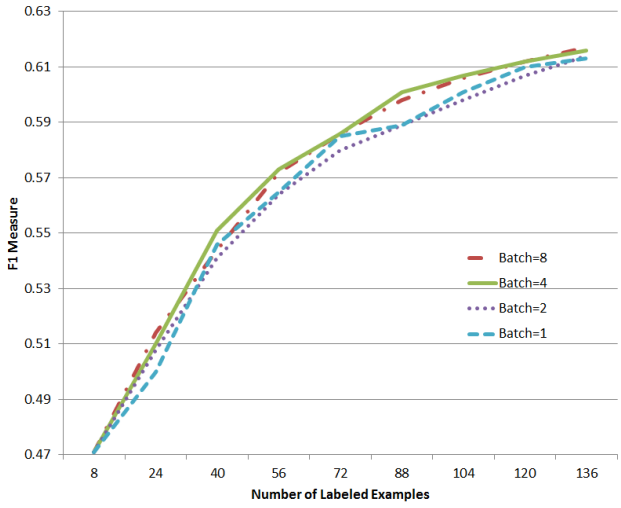

Figure 10: Effect of Batch Size

To this end, we chose Naïve based active learning with entropy based sampling. We varied the batch size from 1, 2, 4 and 8 and ran the experiment for 50 runs. The plot can be seen in Figure 10. As the plot suggests, the performances are less sensitive to batch sizes. A reasonable choice could be a batch size of 4. But again, it depends on the amount of computation cost available for model construction.

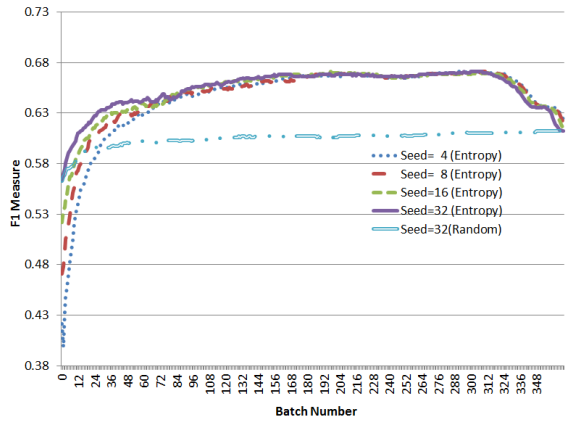

Figure 11: Effect of seed data

In the last experiment, we varied initial seed size to see its effect of the initial seed size on our active learning approach. We experimented with seed sizes of 4, 8, 16 and 32. We applied Naïve based active learning with the batch size of 4 and 100 runs. The plot in Figure 11 shows F1 measures of Entropy based sampling at different seed set sizes. It can be seen that the smaller the seed size, the smaller the F1 score initially. Having a larger seed data initially is beneficial which is obvious because in general the larger the training set the better. We also included a plot of the F1 measure corresponding to random sampling with 32 seeds in Figure 11. It is interesting to note that although random sampling with 32 seeds has larger F1 score initially, it eventually performs poorly when more data is added.

6 Conclusion

In this paper, we proposed to use active learning for training classifiers for judging the quality of automatically generated gap-fill questions, which is the first attempt of its kind to the best of our knowledge. Our experiments showed that active learning is very useful for creating cost-efficient methods for training question quality classifiers. For instance, it is observed that a reasonably good classifier can be built with 300-500 labeled examples using active learning (a potential stopping criteria) that can provide about 5-10% more in accuracy and about 3-5% more in F1-measure than with random sampling. Indeed, the proposed approach can accelerate the question generation process, saving annotation time and budget.

Although the proposed method is investigated in the context of judging the quality of gap-fill questions, the method is general and can be applied to other types of questions e.g., stem generation for multiple choice questions and ranking of free-response questions. We plan implement the remaining steps (i.e. sentence selection and candidate generation) of the question generation pipeline and make it a complete system.

Acknowledgments

This research was partially sponsored by The University of Memphis and the Institute for Education Sciences (IES) under award R305A100875 to Dr. Vasile Rus.

References

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

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

Jinying Chen, Andrew Schein, Lyle Ungar, and Martha Palmer. 2006. An empirical study of the behavior of active learning for word sense disambiguation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 120–127. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Sarah H Eason, Lindsay F Goldberg, Katherine M Young, Megan C Geist, and Laurie E Cutting. 2012. Reader–text interactions: How differential text and question types influence cognitive skills needed for reading comprehension. *Journal of educational psychology*, 104(3):515.

Arthur C Graesser, Shulan Lu, George Tanner Jackson, Heather Hite Mitchell, Mathew Ventura, Andrew Olney, and Max M Louwerse. 2004. Autotutor: A tutor with dialogue in natural language. *Behavior Research Methods, Instruments, & Computers*, 36(2):180–192.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Michael Heilman and Noah A Smith. 2009. Question generation via overgenerating transformations and ranking. Technical report, DTIC Document.

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

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

Ming Liu, Rafael A Calvo, and Vasile Rus. 2012. G-asks: An intelligent automatic question generation system for academic writing support. *Dialogue & Discourse*, 3(2):101–124.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Karen Mazidi and Rodney D Nielsen. 2014a. Linguistic considerations in automatic question generation. In *Proceedings of Association for Computational Linguistics*, pages 321–326.

Karen Mazidi and Rodney D Nielsen. 2014b. Pedagogical evaluation of automatically generated questions. In *Intelligent Tutoring Systems*, pages 294–299. Springer.

Rada Mihalcea. 2004. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 20. Association for Computational Linguistics.

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.

Nobal B Niraula, Vasile Rus, Dan Stefanescu, and Arthur C Graesser. 2014. Mining gap-fill questions from tutorial dialogues. pages 265–268.

MJ Pollock, CD Whittington, and GF Doughty. 2000. Evaluating the costs and benefits of changing to caa. In *Proceedings of the 4th CAA Conference*.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. 2007. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of the Linguistic Annotation Workshop*, pages 101–108. Association for Computational Linguistics.

Vasile Rus and Arthur C Graesser. 2009. The question generation shared task and evaluation challenge. In *The University of Memphis. National Science Foundation*.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 251–257. Association for Computational Linguistics.

Vasile Rus, Nobal Niraula, Mihai Lintean, Rajendra Banjade, Dan Stefanescu, and William Baggett. 2013. Recommendations for the generalized intelligent framework for tutoring based on the development of the deeptutor tutoring service. In *AIED 2013 Workshops Proceedings*, volume 7, page 116.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 589. Association for Computational Linguistics.

Jason D Williams, Nobal B Niraula, Pradeep Dasigi, Aparna Lakshmiratan, Carlos Garcia Jurado Suarez, Mouni Reddy, and Geoff Zweig. 2015. Rapidly scaling dialog systems with interactive learning.