AUTOMATIC QUESTION GENERATION AND STUDENT ANSWER ASSESSMENT IN DIALOGUE-BASED INTELLIGENT TUTORING SYSTEMS

by

Nobal Bikram Niraula

A Dissertation

Submitted in Partial Fulfillment of the

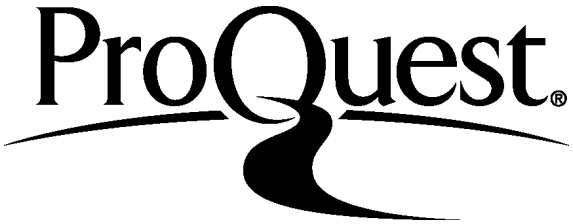
Requirements for the Degree of

Doctor of Philosophy

Major: Computer Science

The University of Memphis

August 2015



DEDICATION

To my parents, my brother and my wife.

ACKNOWLEDGMENTS

I am very much thankful to my advisor, Dr. Vasile Rus, for his guidance and great support to my research and professional activities. His supports have influenced me in my development as a research scientist. I am really grateful to Dr. Dipankar Dasgupta for his initial support to my PhD and constant guidance towards my degree. I also appreciate the rest of my dissertation committee members Dr. Arthur C. Graesser, Dr. Lan Wang, and Dr. Scott D. Fleming for their helpful comments and thoughtful discussions.

I would like to acknowledge the financial supports from the grant R305A100875 from the Institute for Education Sciences to Dr. Vasile Rus, the Institute for Intelligent Systems, the Institute for Intelligent Systems Student Organization, and the Graduate Student Association of the University of Memphis.

Furthermore, I am grateful to my co-authors and my friends including Archana Bhattarai, Rajendra Banjade, Mihai Lintean, Dan Stefanescu, Nabin Maharjan, Dipesh Gautam, Vivek Datla, Brent Morgan, and William Baggett. I strongly value their supports for data annotation and participation in fruitful discussions to my research activities.

I would like to thank my brother Prakash B. Niraula, my parents, my sisters and my in-laws. I would not be at this stage I am today without their constant support and great sacrifices towards my education along the way. I truly thank Mulibir Rai, Kesar Rai, and Chakramani Rai for their great financial and educational support that paved the way for my higher studies.

My special thanks go to my wife Diwa Koirala for her unflagging support and patience. Words are not enough to express her encouragement and love in my life. Thank you very much for always being there for me. Finally, I would like to thank my son, Nishan for his sweet smile when I needed it.

ABSTRACT

Niraula, Nobal Bikram Ph.D. The University of Memphis. August, 2015. Automatic Question Generation and Student Answer Assessment in Dialogue-based Intelligent Tutoring Systems. Major Professor: Vasile Rus, Ph.D.

Dialogue-based Intelligent Tutoring Systems (ITSs) have already proven to be very effective at inducing learning gains in students. These systems are guided by dialog scripts, the heart of many dialog systems, for the interactions with students. The scripts typically consist of a list of questions and corresponding ideal answers. In most ITSs, such scripts are manually crafted from instructional task descriptions. Such manual efforts not only cost more in terms of time and effort but also set a bottleneck in the scalability of the systems. Another major challenge they face is to automatically assess student answers with respect to the ideal answers. To address these challenges, this research proposes novel approaches to automatically generate questions. Furthermore, it focuses on finding appropriate approaches to assess and understand student responses in the form of natural text inputs.

The question generation process generates cloze and open-cloze questions. Cloze questions are automatically generated by mining recorded tutorial dialogues between actual students and a state-of-the-art ITS. It complements the existing systems that rely only on the contents of instructional texts. Open-cloze questions are generated by minimizing human efforts. Specifically, active learning is used to train classifiers for judging the quality of automatically generated open-cloze questions, the most expensive step in generating open-cloze questions. Experiments show that a reasonably good classifier can be built with 300-500 examples labeled by using active learning which can provide about 5-10% more in accuracy and about 3-5% more in F1-measure than random sampling.

Towards assessing and understanding student responses, this research addresses pronoun resolution and semantic textual similarity (STS) problems in the context of tutorial dialog. For pronoun resolution, a supervised machine learning

approach is proposed which has a F-measure of 88.93%, showing its robustness in resolving pronouns. For assessing student responses, STS methods are used as they provide numeric scores indicating the degrees of equivalence in meaning between student answers and corresponding ideal answers. Since student responses in tutorial dialog are typically short in length, this research seeks to find the best methods for short text-to-text STS. To this end, Latent Dirichlet Allocation-based and regression-based methods are proposed. The methods are found to be very promising for computing short text-to-text semantic similarities.

Although approaches to the STS problem provide numeric scores, they fail to explain the reasons behind them. To this direction, an interpretable STS system has been proposed which has been ranked at the top tier of this kind in the literature.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Intelligent Tutoring Systems

Intelligent Tutoring Systems (ITSs) are relatively new Natural Language Processing (NLP) applications to education. They belong to a category of advanced educational technology that tailor instruction to each individual student in order to maximize learning for every single student. Indeed, ITSs have already proven to be very effective at inducing learning gains in students (Rus, D'Mello, Hu, & Graesser, 2013; VanLehn, 2011a). For instance, VanLehn (2011a) reported an effect size (the performance measure for learning gains) of 0.76 for ITSs which is very close to the effect size of human tutors (0.79).

The nature of interaction with an ITS varies from multiple choice questions to fully dialogue based. A lot of research works are ongoing towards developing effective computational models from computational linguistics, artificial intelligence, psychology, and other interdisciplinary fields. Many of the ITSs that have developed during the last 20 years have proven to be quite successful, particularly in the domains of mathematics, science, and technology (Graesser, VanLehn, Rosé, Jordan, & Harter, 2001). Autotutor (Graesser et al., 2004) and DeepTutor (Rus, Stefanescu, et al., 2014) are two typical examples of dialogue-based ITSs for teaching conceptual Physics.

An ITS experiment typically consists of *pretest*, *learning* and *posttest*. The *pretest* measures the existing knowledge (i.e., the pre-knowledge) that a student has before the tutoring. The score also helps the tutor to choose the appropriate challenges (aka. *tasks* or *problems*) for the student to solve during the tutoring session. The *learning* involves solving a number of *tasks* by a student with a series of dialog interactions with an ITS. Finally, the student takes *posttest* to evaluate the

effectiveness of learning (of an ITS) which can be measured in terms of *learning gains*.

VanLehn (2011b) proposed a general architecture for an ITS which is presented in Fig. 1.1. According to this architecture an ITS consists of two loops: the *outer loop* and the *inner loop*. The *outer loop* helps to determine the appropriate tasks and their ordering for a student based on the principle of learning undertaken by the ITS. The *inner loop*, on the other hand, provides adaptive feedback while solving a problem. Specifically, in the inner loop, tutor asks a question and the student submits the solution to the question. The tutor then assesses the student's answer to determine whether the task is completed. If the task is achieved, then the inner loop handovers the control to the outer loop. Otherwise, the inner loop provides feedback and hints to the student for achieving the task.

Although ITSs were initially emerged from academics where the major goal was to help students learning the instruction materials, their encouraging learning gains pushed them beyond the academic settings. For example, industries have started developing ITSs to train their employees for the internal processes, new tools and techniques etc. as they reduce the training cost (Ong & Ramachandran, 2003). In addition, with personalized ITSs, employees can learn the contents at their own pace and time, providing a great flexibility over human instructor-based training.

1.2 Research Challenges and Possibilities

In this electronic era, digital contents such as Wikipedia[1] and free online books, essays, and articles are wide-spread over the Internet. That is, learning materials are abundant and is not a concern for learning. The real concern for learning is the effectiveness of the learning. It is reported that reading learning materials without any tutor is less effective than learning the same materials using

[1]http://wikipedia.org

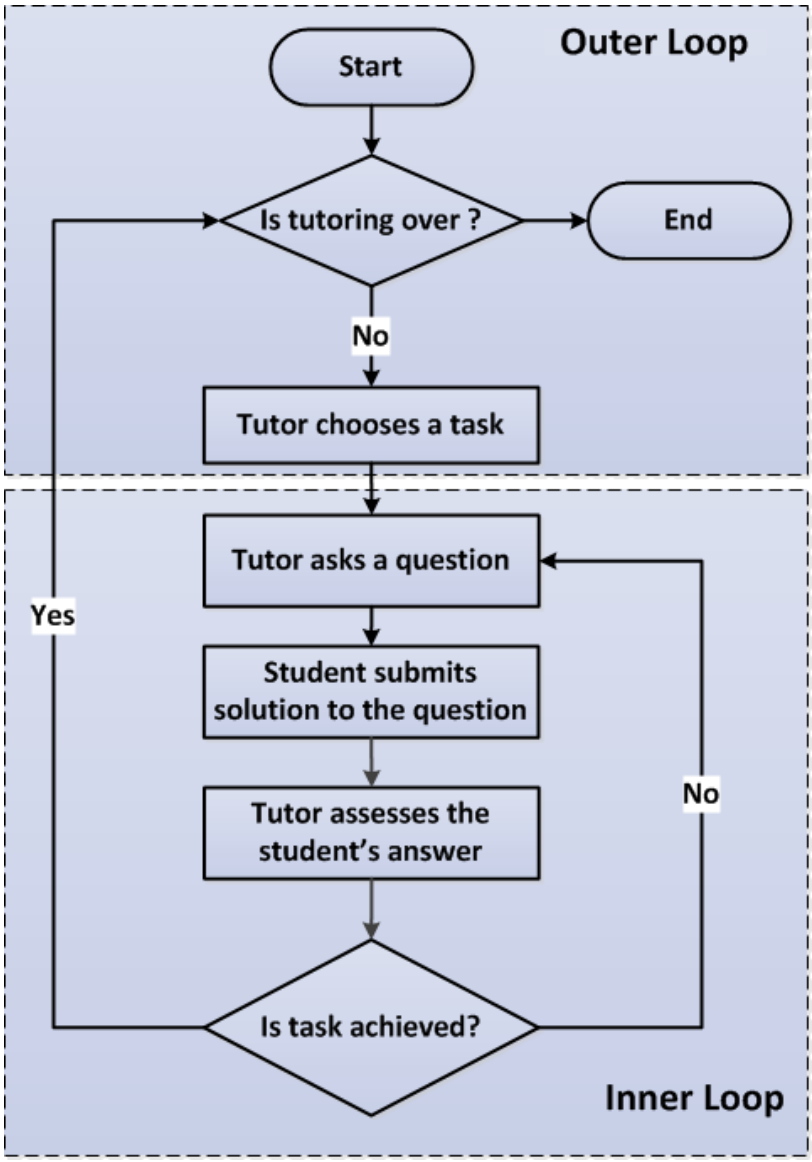


Fig. 1.1: Inner and outer loops in an ITS

ITSs because they provide interactive and adaptive environments to the learner (VanLehn, 2011a).

Digital contents are not readily consumed by the ITSs. In order to do so, the contents have to be processed and transformed to the suitable formats of the ITSs. For dialogue-based ITSs, the processing typically takes learning materials as input and generates dialog scripts, the heart of the dialog systems, for the learner-tutor interactions. In most of these systems such scripts are handcrafted from instructional task descriptions which typically consist of generating a list of problems and the corresponding ideal answers, and feedback and hints in the form of questions. However, manual approaches not only cost more in terms of time and effort but also set a bottleneck in the scalability of the system. For instance, it is estimated that 200 hours of content development are required for creating an hour of instruction (Anderson, Corbett, Koedinger, & Pelletier, 1995). Thus, automatic generation of questions is very useful while building dialog systems such as ITSs. Indeed, it is already proven that test construction is an expensive and a very time-consuming process for instructors and educational researchers and the use of computer assisted assessment dramatically reduces the burden (Pollock, Whittington, & Doughty, 2000). Mitkov, Ha, and Karamanis (2006) also reported that automatic question construction followed by manual correction is more time-efficient than manual construction of the questions alone.

Furthermore, in a conversational ITS it is important to understand students' natural language inputs in order to assess their level of understanding of the target topic to be learned and, consequently, provide appropriate feedback (Rus, D'Mello, Hu, & Graesser, 2013). One frequently used approach to address this student input assessment problem is to compute how similar the student response is to a benchmark response such as an expert-articulated response (Graesser, Olney, Haynes, & Chipman, 2005; Rus & Graesser, 2006). That is, the student response

assessment task is being modeled as a text-to-text similarity problem. It is a practical alternative of finding true understanding of student response which is intractable as it requires world knowledge. As such, automatic and effective methods for text-to-text semantic between student responses and benchmark answers are very critical requirements for scaling and effectiveness of an ITS.

As mentioned above, on one hand we found tremendous research works that are validating the effectiveness of ITSs. On the other hand, we observed question generation and student answer assessment as two key bottlenecks in scaling ITSs. These observations provide motivations for seeking the techniques that address the bottlenecks.

1.3 The Goal

We concentrate on automating the inner loop of a dialogue based ITS rather than focusing on the outer loop. In other words, we assume that we receive a task to be solved by a student from the outer loop. Our goal, for the given task, then is to generate hints in the form of questions and understand student responses to provide appropriate feedback.

Before explaining our approaches, let's examine the real dialogue flows between a state-of-art dialogue-based ITS, DeepTutor, and a student in the inner loop. We presented an snapshot of DeepTutor while it was at its inner loop in Fig. 1.2. Student is challenged to solve a task whose description is given in the top-right block in the picture. Tutor asks a question to the student and the student answers the question by typing his answer to the bottom-left section of the figure. The dialogues interactions are shown in the left-top section of the figure. The right-bottom section provides images that are relevant during the conversation.

The templates for dialog flows in many state-of-art tutoring systems are constructed manually. The process starts with writing a (problem, ideal answers) pair. Several types of questions corresponding to the ideal answers are then

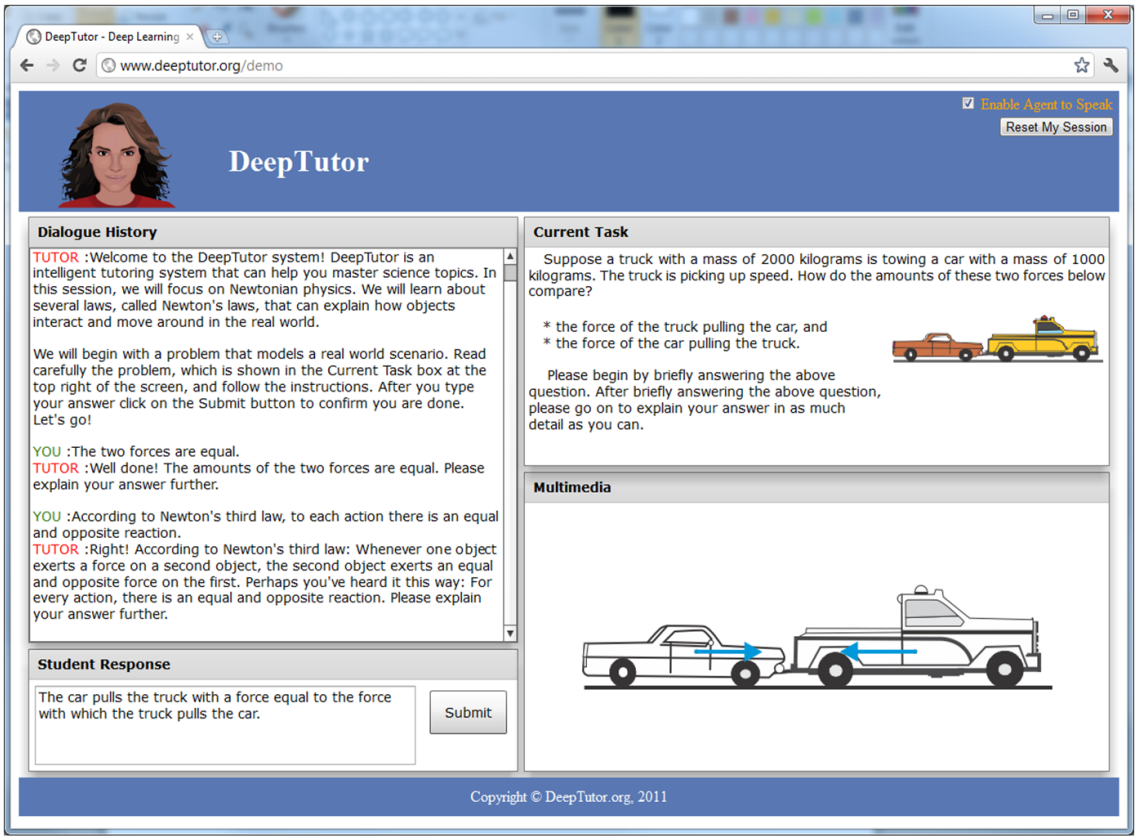


Fig. 1.2: DeepTutor at inner-loop

formulated. When a student submits a response to a question, his answer will be compared with the ideal answer. If the student answers the question correctly, the next question is provided, if any. Otherwise the tutor provides hints to solve the question being asked. Typically the hints consists of prompts (e.g. gap-fill question) targeting the ideal answer.

If we observe carefully, there are two major things that are needed for this type of conversational system to work properly. The first is to generate the questions corresponding to sentences in the ideal answers. Typically, experts generate such questions manually. Secondly, the answer given by the students should be matched with the ideal answers. Computing the semantic similarity between the student's answer and the ideal answer is another crucial factor to consider for the system because this determines the type of feedback we give to the

student. Moreover, failing to understand the student's input correctly prompts to choose an incorrect feedback which makes the student frustrated sometimes to the point of quitting the system. The assessment task is more difficult as students frequently use pronouns (such as it, he, and she) in their answers which refer to some entities in the problem or the the tutor's question. The resolution of the pronoun is needed to better access their answers.

While writing templates for dialog generation using manual approach works for small-scale ITS, it becomes a real challenge when the ITS has to be scaled up. Even for a small-scale ITS, new problems are to be encoded manually. This limits the scalability of the ITS and demands more costs in terms of time and effort. Thus, we consider automatic question generation and student answer assessment as two key bottlenecks for the scalability of an ITS. We will address the bottleneck posed by the manual system of generating questions by automatically generating hints. In addition, we will seek for appropriate semantic similarity approaches that help to better understand the students' responses.

1.4 Dissertation Statement and Primary Contributions

This section presents a thesis statement and main contributions of this dissertation work.

1.4.1 Statement

Automatically generating different types of questions and finding effective methods for assessing students' knowledge are two key aspects of dialog generation in an dialog based ITS. The methods that automate the question generation process and reduce manual efforts and costs will be presented. Moreover, effective measures will be investigated for assessing student answers.

1.4.2 Objectives

While the objective of this dissertation is towards the bigger goal of automatic generation of tutorial dialog, the specific objectives are listed as follows:

1. Generate cloze and open-cloze questions for tutorial dialogue

2. Solve pronoun resolution in tutorial dialogue

3. Identify the best semantic similarity methods for tutorial dialog

1.4.3 Summary of Primary Contributions

From this work, we make following contributions towards automatic generation of tutorial dialog.

In Chapter 2, we present techniques to automatically generate different types of questions for tutorial dialog. Particularly, we generate cloze and open-cloze questions. We contribute many novel question generation techniques to the question generation literature including the one that exploits wisdom-of-students i.e., we generate cloze questions by exploiting the students responses to the open-cloze questions.

In Chapter 3, we describe our approach to handle pronouns in tutorial dialog. Students use pronouns in their answers frequently. Thus the assessment of the students responses needs resolution of the pronouns. Although a plethora of pronoun resolution approaches for written texts are available in the literature, they are not optimal for tutorial dialog as these are specific systems with different assumptions. Our adaptation techniques to resolve pronouns in tutorial dialog will provide novel techniques on the literature of anaphora resolution in NLP.

In Chapter 4, we describe our experiments towards assessing the students' responses. Since the students responses are typically short, e.g., single sentence to few sentences, we contribute to the literature on semantic similarity between two short texts in NLP. We also present our approach for interpretable semantic textual similarity that aims at providing reasons behind two texts being similar, related or unrelated.

1.5 Delimitations and Assumptions

Our approach is not a one-size-fits-all solution for intelligent tutoring systems. We have made some specific assumptions which are listed below.

- Dialog-based Interaction: We assume that the interactions between students and the tutor are a chat-like interactions. It means that only the textual inputs are accepted.

- Expectation and Misconception Tailored Model: Although the proposed approach can be used in other models, we inherently support expectation and misconception tailored dialog model as in AutoTutor (Graesser et al., 2004). An expectation refers to a set of sentences representing good answers whereas a misconception refers to a set of erroneous sentences that a tutor anticipates from a learner.

- Conceptual Domains: We assume that our target domain be more conceptual rather than quantitative which requires mathematical computation.

- Input as Problem-Answer Pairs : We assume that experts provide problem-answer pairs from which the tutorial dialog is to be generated. Moreover, deep questions are assumed to be provided by experts.

Chapter 2

Question Generation

An important issue for automatically generating tutorial dialog is to automate the generation of pedagogically good questions by taking the dialog context into consideration. The minimum the cost of the automation, the better the solution. This chapter addresses these issues in detail based on the following published works: Niraula and Rus (2015) and Niraula, Rus, Stefanescu, and Graesser (2014).

2.1 Introduction

Questions are one of the critical components of ITSs. If adaptive to individual learners, they must assess learners' knowledge before, during, and after students' interactions with the platform. For instance, in order to identify knowledge deficits before and after a session, pre-test and post-test can be used respectively. The knowledge deficits discovered based on pre-test can guide the ITSs to select appropriate instructional tasks for the learner. Furthermore, pre- and post-tests can be used to measure learning gains with ITSs, e.g., by subtracting 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-ended questions.

In addition to pre- and post-learning assessments, ITSs use questions to interact with students during learning sessions. For example, DeepTutor starts its inner loop by asking a deep (e.g., why and what) question to a student. The deep question demands not only the answer but its justification too. When the student answers the question, DeepTutor chooses another question based on the assessment of the student's answer. The question could be another deep question or a shallow (e.g., gap-fill) question. The process continues until the task is done.

Besides test construction for ITSs, automatic question generation (QG) are very useful in several other applications such as reading comprehension (Eason, Goldberg, Young, Geist, & Cutting, 2012), vocabulary assessment (Brown, Frishkoff, & Eskenazi, 2005), and academic writing (Liu, Calvo, & Rus, 2012). Other usages include generating questions for frequently asked questions, generating suggested questions for patients and caretakers in medicine domain, generating suggesting question that learner might ask while reading documents and other media etc. (Rus & Graesser, 2009).

Given the wide usage, automatic question generation is very useful while building dialog systems such as ITSs. They not only speed up the scaling process but also lower the human efforts. Indeed, it is already proven that test construction is an expensive and a very time-consuming process for instructors and educational researchers and the use of computer assisted assessment dramatically reduces the burden (Pollock et al., 2000). Mitkov et al. (2006) also reported that automatic question construction followed by manual correction is more time-efficient than manual construction of the questions alone. As a result, a particular attention has been given by the NLP community to automatically generate several types of questions. In fact, automatic question generation for educational purposes is going on over three decades. Autoquest is one of the earliest works which generated questions for independent study of text articles (Wolfe, 1976). Other examples include *factoid question* and *gap-fill questions* (Ali, Chali, & Hasan, 2010; Curto, Mendes, & Coheur, 2011; Heilman & Smith, 2009; Kalady, Elikkottil, & Das, 2010; Mannem, Prasad, & Joshi, 2010; Mitkov et al., 2006; Mostow & Chen, 2009; Varga & Ha, 2010; Wyse & Piwek, 2009; W. Chen, Aist, & Mostow, 2009; Wolfe, 1976; Yao & Zhang, 2010). Moreover, significant contributions have been made to this end via shared tasks (Rus et al., 2010).

Despite its benefits, automatic 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.

2.2 Types of Questions

There are several types of questions, and the literature contains a various taxonomies for organizing them. One of such works is by Graesser, Rus, and Cai (2008) which presents a concise description of the dimensions to classify the questions. They mention five types of characteristics that can be used to categorize the questions. The characteristics are: purpose, type of information, source of information, and length of the expected answer, cognitive processes.

2.2.1 Purpose

Graesser et al. (2008) listed the purpose of questions as: the correction of knowledge deficits, the monitoring of common ground, the social coordination of action and the control of conversation and attention.

2.2.2 Type

Graesser et al. (2008) proposed 16 categories for questions based on the type of information that ranges from simple to complex. Concept completion, verification, goal orientation and judgmental questions are typical examples of these question categories. A system that generates complex question likely requires a significant human knowledge to be encoded for the question type and the domain.

2.2.3 Source

Questions can be classified based on the location of their answers. That is, some questions' answers might lie at some particular sentences whereas some questions' answer might lie at paragraph level, document level or even needs inference, common sense or opinions.

2.2.4 Length of Expected Answer

The questions can be classified based on the expected length of their answers. For instance, some questions require long essays while some other need just a word or phrase.

2.2.5 Cognitive Process

Some question needs simply recognition and recall of information where as some other need the comprehension, inference, application, synthesis or other complex processing is involved.

2.3 Approaches

Automatic question generation for educational purposes is going on over three decades. Autoquest system is one of the earliest works which generated questions for independent study of text articles (Wolfe, 1976). The system was used to generate Wh-Questions from randomly selected sentences from the text articles and was fully based on the syntactic structure of sentences rather than semantic. Several question generation systems are proposed since then which can be broadly classified into three categories: template-based, syntax-based, and semantics-based.

2.3.1 Syntax-based

Syntax-based methods share a large portion of the current QG literature (Ali et al., 2010; Kalady et al., 2010; Varga & Ha, 2010; Wolfe, 1976). The common approach of these systems is to parse the given sentence to find its syntactic structure (i.e., syntax trees). The structure is then exploited to simplify the sentence, to identify the key phrases, and to apply syntactic transformation rules and question word replacement.

2.3.2 Semantic-based

Semantics-based question generation systems apply several transformations to generate questions (Mannem et al., 2010; Yao & Zhang, 2010). As name suggests, the transformations are semantic rather than syntactic. Typically, they use the

information from Semantic Role Labeling to get predicates and arguments and use them to find target words and apply semantic transformations.

2.3.3 Template-based

Template-based approach is probably the most easiest way to generate questions. Systems that rely on this approach use question templates which are pre-defined texts containing placeholder variables. The values of placeholder variables are changed to have different questions. This approach is quite popular in the literature (W. Chen et al., 2009; Curto et al., 2011; Mostow & Chen, 2009; Wyse & Piwek, 2009). Mark-up languages, such as NLGML (Chai et al., 2001), are proposed to facilitate template-based question generation. Although template-based systems easy to build, the major hurdles are that they are domain-specific and need extra human efforts to make such systems viable.

2.3.4 Overgenerate-and-Rank

Overgenerate-and-rank is one of the popular approaches for question generation regardless of question types (Heilman & Smith, 2010; Becker, Basu, & Vanderwende, 2012). The idea is first to generate many candidate questions (overgeneration) and then rank them to obtain the good questions at the top. This approach has been adopted in this work to generate different types of questions.

2.4 Related Works

The target of this work is to generate factoid questions for tutorial dialog which are the questions that can be generated from given sentences. Particularly, the focus will be on generating *gap-fill* and *free-response* questions.

2.4.1 Gap-fill Questions

Gap-fill questions are *fill-in-the-blank* questions which consist of one or more gaps (blanks) in a sentence / paragraph and a number of choices for each gap. This type of questions can be of two types: with choices and without choices. The former

is known as *cloze* (aka multiple choice) question and the latter is know as *open-cloze* question.

One of the choices in *cloze* question is the correct answer to the question, called the *key*. The rest of the choices are the *distractors*, i.e., incorrect answers that tempt less proficient students who often confuse them with the *key*. The sentence containing a gap (s) is also known as the *stem*. Consider a *cloze* question 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.

(a) third

(b) second

(c) first

(d) heating

In the cloze question above, the question sentence contains a gap and there are four potential choices for the gap. The *key* is *second* and *first*, *third* and *heating* are three distractors. Two of distractors are very close to the key while another, *heating* is quite remotely related.

Open-cloze questions are also a type of *fill-in-the-blank* questions consisting of a sentence/paragraph with one or more gaps (blanks). A typical open-cloze 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 open-cloze question presented above has a word missing (i.e., a gap). A open-cloze question can have one more than one gaps too. Students (test takers) are supposed to predict the missing word(s) in their answer(s).

The attractiveness of gap-fill questions is that they are well-suited for automatic grading. This is because the correct answer for open-cloze questions is simply the original word/phrase corresponding to the gap in the original sentence.

The correct answer in case of a cloze question is one of the choices shown to learners. As a result, they are frequently used in educational contexts such as MOOCs and ITSs (Graesser et al., 2003). Specifically, ITSs use such questions for diagnosing and assessing students' knowledge level and their learning gains as parts of their assessment and practice modules.

Related Works in Cloze Question Generation

Four main steps are needed to generate gap-fill questions with choices from an instructive text:

1. Selecting useful sentences from the text
2. Identifying gaps (i.e., words to be deleted) in the selected sentences
3. Generating distractor candidate list and
4. Ranking the distractors in the list

The literature of gap-fill question generation contains methods that go through each of the steps or focus on particular steps. Since we mine the gap-fill questions using the responses given to the open-cloze questions from ITS dialogues, we do not need to address the first two steps of the process.

Mitkov et al. (2006) proposed a computer-aided procedure to generate multiple-choice questions from textbooks that goes through all the four steps. It starts by finding domain specific terms (nouns and noun phrases) using a shallow parser that satisfies certain regular expressions. The terms satisfying a count threshold are considered as key terms. The declarative sentences corresponding to the key terms are converted to WH-questions to form question sentences (aka. stems) using syntactic transformations. Then, they collect hypernyms and coordinates (concepts with the same hypernym) of a term from WordNet (Miller, 1995) to generate the distractor list. The ranking is done using semantic similarity

functions on the assumption that a distractor should be as semantically close to the key as possible.

A more recent work to generate cloze question was proposed by Agarwal and Mannem (2011). Their system automatically generates gap-fill questions from textbooks for reading comprehension tests. Their approach also goes through all four steps of a typical gap-fill question generation process: first, it selects informative sentences from the book using a list of features; second, it finds the key words in each selected sentences using POS tags and chunks of the sentences; third, it generates distractors using contextual similarity (between the key and a candidate word), sentence similarity (between the key's sentence and the candidate word's sentence) and difference between term frequencies (between the key and the candidate word); fourth, it ranks the distractors based on the similarity scores.

Hoshino and Nakagawa (2005) modeled the problem of generating multiple-choice questions as a learning problem. They proposed a machine learning approach to generate such questions for language testing. They did not transform a declarative sentence into a interrogative sentence to form a question sentence rather proposed methods to decide the position of the gap i.e., missing word(s), and corresponding distractors. To decide whether a given word can be left blank 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 blanks positions in the question from the book whereas the negative examples were generated by shifting the blank position. The distractors were the random words from the same article excluding punctuation and the same word.

Sumita, Sugaya, and Yamamoto (2005) generated gap-fill questions considering verbs as gaps in a sentence. Thesaurus was used to obtain distractors for the keys of the gaps. To rank distractors, they took each distractor, filled the

gap using it, and searched the web to get the hit counts of the sentence. The basic idea was that the web-hits for a question sentence when filled by distractors should have low or zero-hits as distractors would lead to incorrect sentences.

There are other works in cloze question generation. For example, Pino, Heilman, and Eskenazi (2008) described a system that started with sample sentences from WordNet to form question sentences as a baseline. They improved the technique using linguistic features. Similarly, Smith, Avinesh, and Kilgarriff (2010) generated cloze questions in English language learning. They used distributional thesaurus to find the distractors.

As one may note, existing cloze and open-cloze question generation approaches were mainly proposed for language learning assessment, vocabulary assessment, and reading comprehension tasks. Most of these works require instructional texts such as textbook chapters and encyclopedia entries in addition to thesauri to generate gap-fill questions. Our approach is novel in exploiting actual student-generated potential distractors from recorded tutorial dialogues.

Related Works in Open-cloze Question Generation

The methods that generate open-cloze questions can be classified into two categories: *heuristics based* and *machine learning based*. Heuristics based methods make some heuristics (e.g., POS of a word) to make a gap. The approach by Sumita et al. (2005) is an example of heuristic based approach. It generates gap-fill questions considering verbs as the gaps in a sentence. Simply considering verbs as gaps does not always work as not all verbs are equally important in a sentence. For example, some missing verbs are easier to guess from the context than other verbs. Moreover, it does not consider other content words such as nouns and adjectives as potential candidates for the gaps. It also does not rank the question candidates.

Another example of heuristics based approach is by Agarwal and Mannem who proposed a system to generate cloze questions (Agarwal & Mannem, 2011)

from text books for reading comprehension tests. As mentioned previously, they first select informative sentences from the book using a list of features and consider the gaps as the key words in each selected sentences by using POS tags and chunks of the sentences.

Machine learning based approaches model the problem of generating questions as a learning problem. Hoshino and Nakagawa (2005) proposed a machine learning approach to generate multiple-choice questions for language testing. They form 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., 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 blanks positions in the question from the book whereas the negative examples were generated by shifting the blank position.

Recently a machine learning based approach is proposed by Becker et al. (2012). Their system, aka. *Mind the Gap*, uses text summarization technique to select useful sentences from text articles for which gap-fill questions are to be generated. For each of the selected sentence, it generates potential gap-fill candidates using semantic constraints. Each candidate question is then labeled to one of *Good*, *Bad* and *Okay* class with the help of Amazon's Mechanical Turkers. They defined *Good* questions as the questions that test key concepts from the sentence and are reasonable to answer, *Okay* questions as the questions that test the key concepts but are difficult to answer (e.g., too long, ambiguous) and *Bad* questions as the questions that ask about unimportant aspect of the sentence or their answers are easy to guess from the context. The data set is then used to build a classifier which can rank the candidate questions with confidence score. They reported that the classifier largely agrees with the human judgement on question quality.

2.5 Cloze Question Generation

Cloze questions are very useful for assessing students' knowledge. In fact, it is reported that knowledge tests with cloze questions are effective at diagnosing and assessing students' knowledge (Mitkov, Ha, Varga, & Rello, 2009). Since this type of questions is very suitable for automatic marking, is has been popular in ITSs for diagnosing and assessing students' knowledge level and their learning gains as parts of their assessment and practice modules (Graesser et al., 2003). This section presents a novel method that mines *cloze* questions from tutorial dialogues.

Table 2.1 shows a real student-tutor interaction from the intelligent tutoring system DeepTutor (Rus et al., 2013). The dialogue is started by the tutor describing a problem followed by an invitation to the student to solve the problem. The solution to the problem must include a justification including concepts and principles of the target domain, which in the case of DeepTutor is conceptual Newtonian Physics. The tutor assesses students' answers and provides appropriate feedback. If some of the steps in the solution are missing, e.g., in case a student did not provide a justification of her answer or some important parts of the justification are vague, the computer tutor will provide appropriate hints, some in the form of open-cloze questions, to help the student articulate the missing or vague parts by herself. Open-cloze questions are frequently used by state-of-art ITSs to provide prompts to the students and guide them towards the learning goals (Graesser et al., 2004; Rus, D'Mello, et al., 2013). We can see an *open-cloze* question posed by the tutor at the end of the dialogue shown in Table 2.1. The student then provided his best answer words to fill the gaps.

In recent online ITSs which offer tutoring to hundreds of students (Rus, Stefanescu, et al., 2014), instructional *tasks or problems* are solved by many students. Consequently, many student responses to hints in the form of questions, some of which are *open-cloze* questions, are collected and recorded in log files. The

Table 2.1: Student-DeepTutor interactions while solving a *task*

TUTOR: A stuntman must drop from a helicopter onto a target on the roof of a moving train. The plan is for the helicopter to hover over the train, matching the train's constant speed before the stuntman drops.
TUTOR: Where should the helicopter be positioned relative to the target? Please begin by briefly answering the above question. After briefly answering the above question, please go on to explain your answer in as much detail as you can.
STUDENT: *ahead of the target*
TUTOR: Let me try again. Which principle can be applied when the motion of an object is complex, for instance, it can be thought of as motion in two perpendicular dimensions?
STUDENT: *Netwon's first law*
TUTOR: Let me give you a hint. The decomposition principle says that the analyses of forces and motion along two _____ directions, such as horizontal and vertical, can be done _____.
STUDENT: *perpendicular, separately*

approach presented here exploits this richness of information available in recorded tutorial dialogues from massive online training with ITSs. An advantage of mining these tutorial dialogues is the fact that we have access to actual students answers to *open-cloze* questions. That is, students' responses to these questions are words that they think best fill the gaps in the *open-cloze* question. Because not all the responses are correct, the incorrect responses can be used as potential candidates for distractors. These candidates are then ranked to obtain the best set of distractors.

The proposed method is unique because it is based on a generative approach, i.e., the potential distractors are generated by students themselves, unlike existing approaches which are discriminative, i.e., they rely on extracting questions from instructional texts. Thus, it complements existing works by mining questions and distractors from recorded dialogues. Furthermore, this is one of the first approaches that relies on actual students answers to create test materials - *wisdom of students*.

2.5.1 The Methodology

As described in the previous section, four main steps are needed to generate gap-fill questions from an instructional text: *selecting* useful sentences from the text, *identifying* gaps, *generating* distractor candidates and *ranking* the distractors. Since the process does not start with content-related text but with student responses to open-cloze questions, the first two steps are not needed. Therefore, this section describes only the processes of finding and ranking of distractors.

Although an open-cloze question can have multiple gaps, this study only considers generating questions with single gap as this is sufficient for a proof of concept. Moreover, the methodology for single gap questions could be easily extended to handle cloze-questions with multiple gaps.

Generating Distractor Candidates

Finding plausible distractors that separate knowledgeable students from knowledge-poor students is one of the major challenges for cloze question generation. A good distractor is a concept that is semantically similar at some extent to the key but it is not a correct answer (Mitkov et al., 2006).

Many existing approaches to finding distractors use WordNet (Mitkov et al., 2006; Pino et al., 2008), in-house thesauri (Smith et al., 2010; Sumita et al., 2005), or words from instructional texts (Agarwal & Mannem, 2011; Hoshino & Nakagawa, 2005). In contrast, the current approach uses a generative approach to find the distractors meaning that it exploits responses generated by students to open-cloze questions during tutorial dialogues. The hypothesis is that the incorrect responses given by students are a good source of distractors because these are actual words that students thought would best fill the gaps.

Because open-cloze questions are answered by many students in massive online ITSs, or massive online courses (MOOCs) for that matter, there is a large pool of candidate distractors from which to select. To exemplify this, consider an

example extracted from a tutorial dialogue corpus of a state-of-art ITS in Table 2.2. The open-cloze question contains a gap and is solved by 15 students. Altogether there are 9 different answers which are ranked in descending order by their frequency. The word *curved*, which is the key of the problem, is the provided by 4 students. Similarly, the word *straight* is the answer provided by three students and so on.

Note that not all open-cloze questions used as hints are equally triggerred while students solve problems with a DeepTutor. This happens because students who correctly solve a problem in their first attempt will simply get positive feedback from the system. There is no need to provide scaffolding in such cases in the form of helpful hints, some of which could be open-cloze questions. That is, high knowledge students who do well at solving problem will most likely not see the open-cloze questions. In contrast, low knowledge students who are less likely to solve a problem immediately without any assistance, would more likely be prompted with open-cloze questions as a scaffolding move. In others words, low knowledge students are more likely to receive hints from the computer tutor in the form of open-cloze questions. This is good news for our approach because it means we will have a good pool of distractors as low knowledge students will more likely provide incorrect answers, i.e., distractors. The bottom line is that some open-cloze questions may not have enough student responses, depending on the distribution of students as low or high knowledge students, from which to select distractors. In such cases, we follow some of the existing techniques for finding distractor candidates, e.g., we use WordNet as in (Mitkov et al., 2006). We extract the hypernyms and coordinated concepts (concepts with the same hypernym) of the key and consider them as the potential distractor candidates for the key.

Table 2.2: Students' responses and their frequencies (i.e., votes) to an open-cloze problem

While the wind is blowing, the shape of the sled's path will be _____.
curved = > 4
straight = > 3
diagonal = > 2
no = > 1
linear = > 1
uhm no = > 1
idk = > 1
a triangle = > 1
west = > 1

Ranking Distractors

Once we have the distractor candidates for a key, the next step is to rank them. The top candidates in the ranked list will be used as distractors for the fill-gap question. The example in Table 2.2 indicates that not all student responses are good candidates for distractors. For instance, *idk*, *no*, *uhm no* etc. are not good distractors compared to *straight*, *diagonal* for the key *triangle*.

We used the following two ideas to rank the candidates:

- *R1*: Use the semantic similarity score between the key and the distractors. This idea was used by Mitkov et al. (2006), according to which, a good distractor is very related but not identical to the key. We used Latent Semantic Analysis (LSA) (Landauer, McNamara, Dennis, & Kintsch, 2013), a fully unsupervised methods for inferring words meaning, to get the similarity score between the key and the distractors. LSA is a vectorial representation in which a word is represented as a vector in a reduced dimensionality space, where each dimension is believed to be representative of an abstract/latent semantic concept. Computing the similarity between two words is equivalent to computing the cosine, i.e., normalized dot product, between the corresponding word vectors. Candidate distractors with high similarity scores

are given higher rank than those with low similarity scores. In the running example, similarity scores between word pairs (*curved*,*idk*), (*curved*,*no*), and (*curved*,*uhm no*) will be lower compared to the similarity scores between the word pairs (*curved*,straight), (*curved*,*diagonal*) and (*curved*,*triangle*). This results in the former candidates being moved at the bottom of the ranked list and the latter at the top rank.

- *R2*: Use the counts. A candidate distractor for a question is the number of students who typed the candidate as the answer to the question. According to this idea, the top three distractors for the problem in Table 2.2 are *straight*, *diagonal*, and *no*. These are the top three non-key words in the list after *curved*, the key of the problem. Whenever there is a tie i.e., two distractors have same counts, we break the tie using the semantic similarity score.

2.5.2 Experiments and Results

We first analyzed the distribution of gaps in the set of open-cloze questions from our tutorial dialogue corpus which consists of recorded interactions between high-school students and the intelligent tutoring system DeepTutor. A total of 113 Physics problems were available for use.

The distribution of gaps in the open-cloze questions associated with these problems are presented in Fig. 2.1. It can be observed that the questions with one and two gaps are the most frequent: 65% of the questions have a single gap and 30% of the questions have two gaps. More gaps reduce the contextual information in the problem text, making it harder for studnts to answer correctly. We only focus on single-gap open-cloze questions in the rest of the experiments. It would be somehow straightforward to extend the current solution to questions with multiple gaps.

Next, we mined the corpus of tutorial dialogues obtained from our two experiments that spanned for several weeks. From the first experiment we extracted tutorial dialogues for 297 students who solved 32 tasks (problems). Since a task was

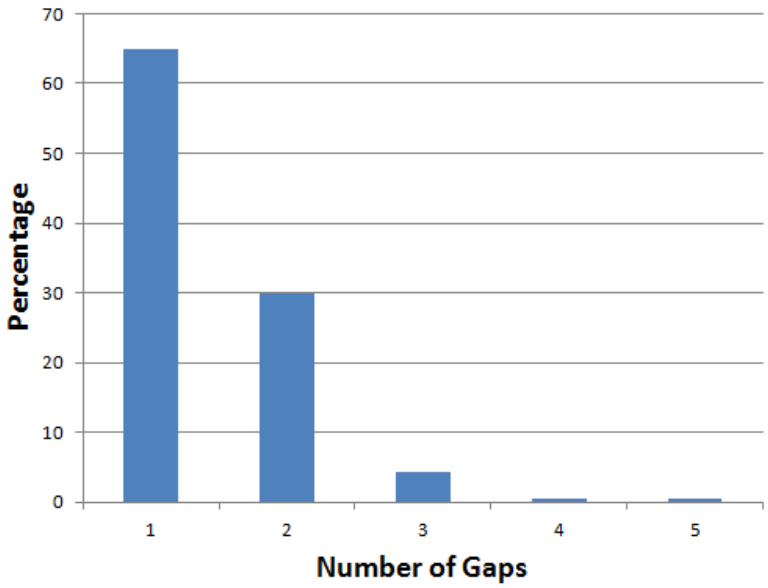


Fig. 2.1: Percentage of gaps in the open-cloze questions

solved by zero or more students, we had 2,687 task sessions altogether i.e., on average 9 tasks per student. Similarly, from the second experiment, we extracted 4,430 task sessions corresponding to 349 students and 37 tasks. That is, on average a student solved 13 tasks in the second experiment.

A total 102 unique single-gap open-cloze questions were mined from the dialogues. We then computed the number of student responses per each of these questions. The result is plotted in Fig. 2.2. It is very interesting to see that some questions received a large number of responses while others received only a few. It would be interesting to see the statistics in weighted counts form. For this, consider Table 2.3 where we provided the percentage of questions corresponding to different count thresholds. All of the single gap open-cloze questions (100%) received at least two responses. Similarly, 82.85% of the problems received at least three responses and 72.38% of questions received at least 4 responses. Interestingly, 44.76% of questions have at least 10 responses. These figures motivate us to generate cloze-questions from the responses of open-cloze questions.

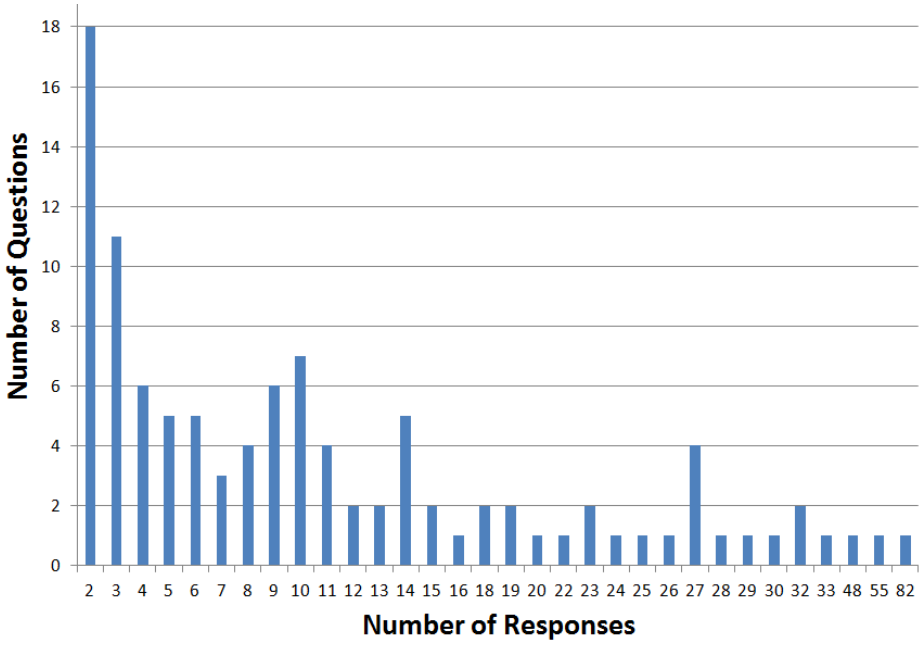


Fig. 2.2: Distribution of questions over number of responses

Relation between a Response's Similarity and its Rank

In this experiment, we wanted to see if there is any relation between the frequency rank (FR) of a student response and the semantic similarity score between the student response and the corresponding key. We define the FR of a student response i in a q question as follows:

$$FR(i) = \frac{100 * f_i}{\sum f_i}$$

where f_i is the number of students who typed i as the answer for q. We call f_i as the votes received by i (or simply the votes of i). The FR of i, thus, is nothing but the percentage of votes i received out of the total votes for q. For instance, FR of the response *curved* in Table 2.2 is 26.6(= 100*4/15) because it has 4 responses out of total 15 responses in the problem.

Table 2.3: Percentage of questions meeting the count threshold

Counts ($\geq$)	2	3	4	5	10
Percentage(%)	100	82.85	72.38	66.66	44.76

We gathered for each open-cloze question its key and the corresponding

student responses. Because many students can give the same response, we computed the frequency and *FR* for each student response. Student responses had to be filtered as they may have contained spelling mistakes, smileys, etc.(see Table 2.2). We detected and discarded such responses using simple rules and a small lexicons of such words or emoticons. Then for each student response, we computed its similarity with the corresponding key. We used LSA to compute the similarity score between each response and the key.

Once we obtained the similarity and *FR* values for all student responses, we computed a correlation coefficient between the scores at different levels of response frequencies. The results are presented in Table 2.4. First, we computed the coefficient for all responses (i.e., minimum frequency ≥ 1) and obtained a correlation coefficient of 0.682. Next, we computed the score for responses generated by at least two students (i.e., minimum frequency $\geq$2) and obtained the correlation coefficient of 0.72. We repeated the process for minimum frequencies of 3, 4, and 5, and obtained correlation coefficients of 0.737, 0.733 and 0.754 respectively.

Table 2.4: Correlation between similarity and weighted frequency

Min Freq	1	2	3	4	5
Correlation_LSA	0.682	0.725	0.737	0.733	0.754

The positive correlation coefficients indicate that there is clearly a positive relation between the rank of a response and its semantic similarity with the key. As we noticed, the correlation coefficients increased as we increased the minimum frequency. Based on these results, we conclude that ranking student responses by their similarity scores with the key is approximated by the vote counts of responses it received from students, i.e., how many students generated the answer. The higher the counts, the more similar the response is to the key. As the distractors for a key

should be as semantically close to the key as possible, we can rank the incorrect responses by their frequencies and utilize them as potential distractors .

Evaluation of Distractor Selection

We conducted two types of evaluations in order to assess the quality of distractors generated by our automated method. In each evaluation, we generated the distractors for the key in a gap-fill question and performed a fine-grain analysis for each distractor to test their quality. Specifically, we classified each of the generated distractors into three categories: *good*, *ok*, and *bad*. The *good* distractors are the ideal distractors, the *ok* distractors can be considered as potential distractors but are not as appropriate as the good distractors. The *bad* distractors do not make sense as a distractor or have the exact meaning with the key.

In the first experiment, we considered questions that had at least three different student responses and had at least two votes per response. There were 23 questions that satisfied this condition. We ranked the distractor candidates by using *R2* presented at Section 2.5.1. That is, we ranked them by their frequencies and chose the top 3 candidates as the three distractors for the question. If two candidates had the same frequency, we ranked them using their semantic similarity score with the key. We rejected the candidates if they were synonyms of the key. We considered a key and distractor synonyms when their semantic similarity score was above or equal to 0.9. We also made sure there were no duplicate distractors in the final list. To reduce the annotation bias, we introduced a random word from a Wikipedia article as the fourth distractor. The order of the four distractors were randomized. Next, we provided annotators the question sentence, its key and asked them to classify each of the four distractors into three categories: *good*, *ok*, and *bad*. A typical annotated instance is showed in the Table 2.5.

We computed the inter-rater agreements using the unweighted version of the Cohen's kappa statistic. The statistic was 0.64 when we considered *good*, *ok*, and

Table 2.5: Sample annotation

Question	The force of gravity exerted by the Earth on the cat is ____ all the time.			
Key	constant			
Distractors	relative	horizontal	zero	smile
Annotation	good	good	good	bad

Table 2.6: Annotation results for 23 questions with 4 distractors each

	good	*ok*	*bad*	*expected bad*
Annotator 1	46	11	35	23
Annotator 2	41	16	35	23

bad groups separately. It increased to 0.86 when we merged *good* and *ok* groups into a single group. The annotation results are presented in Table 2.6. The proportion of the good questions is the highest among all groups for both annotators. Since we introduced one bad distractor per question and we had 23 questions, we expected 23 bad distractors per annotator. Discounting this number in the table clearly indicates that we can achieve very good distractors using the voting scheme. Note that we did not even check the part-of-speech of the distractors when we chose the top 3 candidates as the distractors. This means that it is further possible to boost the performance by considering their parts-of-speech and the higher threshold for the minimum votes.

In the second experiment, we considered 100 single gap questions. We generated three lists of distractor candidates for each question. The first list was generated by ranking the student responses using the *R2* described at Section 2.5.1. It is possible that different questions can have the same key. Thus, we combined all the student responses in the whole dialog corpus for a key and ranked them using the *R1*, i.e., using the semantic similarity scores between the distractor and its key. The motivation here was that we wanted to use as many students responses as distractor candidates as possible. We extracted the distractors from the combined

list corresponding to a key and put them in our second list only if they were not in the first list. For the third list, we extracted the hypernym and coordinates of the key from WordNet as described in Section 2.5.1 and ranked them using their semantic similarity score with the key(*R1*).

We combined all three lists into a single list while preserving their order. This was needed to process the distractor candidates in the first list before the candidates in the second and third lists, and the candidates in second list before the candidates in the third list. For each distractor in the list, we checked whether their parts-of-speech matched with that of the key. If matched, we put the candidate as a potential distractor for the key. Once we had three distractors, we stopped. The forth distractor was generated using a random word from a Wikipedia article as in the first experiment.

The annotation results for this experiment are shown in Table 2.7. The unweighted Cohen's kappa static was 0.54 when we considered all the three classes separately and 0.63 when we considered *good* and *ok* as classes as one. Since we had 100 questions and we introduced a distractor randomly per question, we expected 100 bad distractors from each annotators. Compared to the results in Experiment 1, the concentration of the bad distractors increased and the inter-rater agreement decreased.

Table 2.7: Annotation results for 100 questions with 4 distractors each

	good	*ok*	*bad*	*expected bad*
Annotator 1	161	64	175	100
Annotator 2	122	47	231	100

To identify the causes that degraded the quality of the distractors, we analyzed the distractors that were marked *bad* by the annotators. We noticed an interesting fact about the quality of the distractors that were generated by the

WordNet. Noticed that we used the hypernyms and coordinate concepts of a key to find the potential distractors (see Section 2.5.1) and we ranked them by the similarity scores between the key and the distractors. Although this returned the similar concepts with the key, it failed to provide the distractors that were suitable for the context of the question. An annotated instance shown in Table 2.8 exemplifies the problem. The fourth distractor *black* was a random word inserted in the annotation. The WordNet-based approach generated *one*, *two*, and *three* as the three distractors for the key *zero*. The three candidates are perfect distractors for *zero* when we removed the context. However, for the given question, they are bad distractors. Since students provide contextual words as their answers to the open-cloze questions, considering their responses as distractor candidates would not face this challenge and supports to our hypothesis.

Table 2.8: A problem with WordNet-based distractor selection

Question	Newton's first law says that if an object moves with a constant velocity or is at rest, the net force on the object is _____.
Key	zero
Distractors	one two three black
Annotation	bad bad bad bad

Note that we have many good candidates for distractors which were not selected as we only need three distractors per question. We can utilize such distractors to generate gap-fill questions dynamically. One idea would be to select top N good candidates and choose 3 of them randomly. This would generate different gap-fill questions each time, increasing the diversity of the answer choices and eventually reducing effects of gaming-the-system behavior.

Analysis of Errors

Since our approach relies on student-generated responses, it is prone to errors found in those responses. In fact, this was one of the major factors for limiting the

performance of the system. The first problem was misspelling of words. Students sometimes make spelling mistakes: *seperately* vs *separately*, *thirrd* vs *third* etc. These can be easily repaired at some extent except cases where the mispelled word is itself a valid word, for instance, *on* instead of *no*. In this case, both of these words are valid words unlike in the case of misspelling. Use of numbers is another problem. Students typed numbers in their answers, e.g., 1st for first, 0 for zero, 9.8m/s for constant acceleration etc. They also use phrases for single word keys e.g., *gravity* vs *force of gravity*.

The most challenging factor was finding the similarities between students' answers with the key. Although the words in the following pairs (*is*, *equals*), (*vertical*, *y-direction*), (*identical*, *constant*) may have slightly different meanings, they have same meanings in the context of Newtonian Physics. Our semantic similarity measure, i.e., LSA, failed to handle such pairs and thus performed poorly at finding distractors in those cases. Domain knowledge is required to properly handle these pairs. A domain-specific LSA space might be a good solution, which we plan to do as part of our future work.

2.6 Open-cloze Question Generation

A typical pipeline to automatically generate open-cloze questions is shown in Fig. 2.3. It follows the three steps paradigm for question generation (Rus & Graesser, 2009) : *Sentence Selection*, *Candidate Generation* (overgeneration) and *Candidate Selection* (ranking).

Step 1 - Sentence Selection: To generate open-cloze 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.

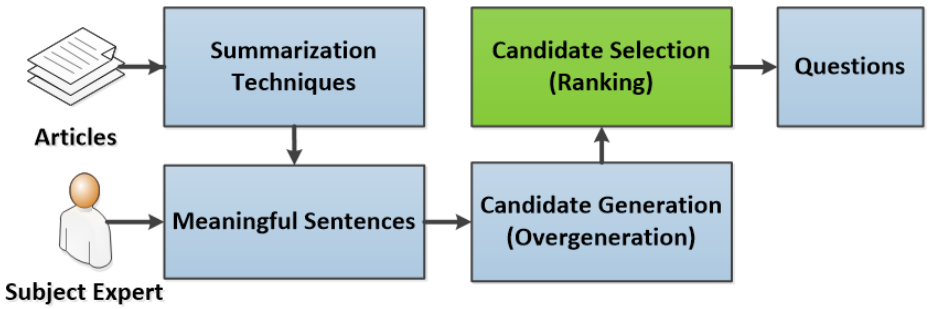


Fig. 2.3: A pipeline for gap-fill question generation

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 2.9.

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

2.6.1 Question Quality

Question quality can be judged linguistically or pedagogically. In linguistic evaluation, questions are evaluated with respect to whether they are grammatically

Table 2.9: Typical overgenerated questions from a sentence with their ratings *Good*, *Okay* and *Bad*.

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.

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 open-cloze 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 & Craven,

2008), word sense disambiguation (J. Chen, Schein, Ungar, & Palmer, 2006), named entity tagging (Shen, Zhang, Su, Zhou, & Tan, 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 2.6.3) 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.

Here, we propose an active learning based approach to judge the quality of open-cloze 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 open-cloze 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 learning approach.

2.6.2 Existing Approaches for Ranking Questions

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 and open-cloze questions. For example, Heilman et al. (Heilman & Smith, 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 (Hoshino & 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 open-cloze questions. They used text summarization technique to select useful sentences from text articles for which open-cloze questions are to be generated. From each of the selected sentence, it generated potential open-cloze 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. That 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 (Mazidi & 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.

2.6.3 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 open-cloze questions.

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.

Algorithm 1: Pool-based active learning algorithm

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

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.

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.

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 & 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 (J. 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 (J. 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.

2.6.4 Experiments

Data Set

Although an active learning system does not 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)[1]. 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 2.6).

For 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.

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 2.10. 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

[1]http://research.microsoft.com/s̃umitb/questiongeneration

Table 2.10: List of features used in the experiments

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)

used 5 different *Token Count* features to capture such properties. We also used 9 *Lexical* features to capture different 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 toolkit (Manning et al., 2014) for getting POS tags and named entities.

Results

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 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.

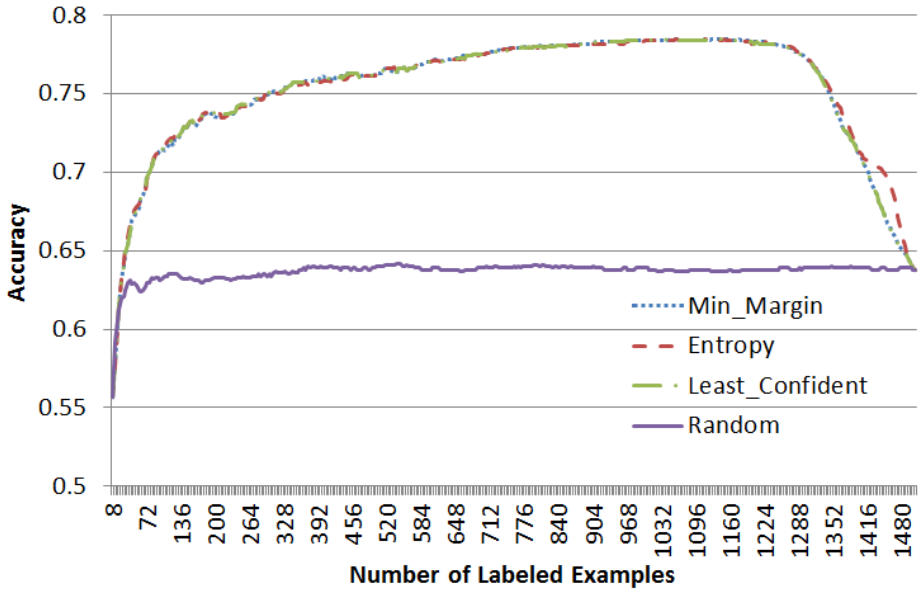


Fig. 2.4: Full simulation for Naïve Bayes accuracy

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. Fig. 2.4 and Fig. 2.6 present the accuracy and F1 scores of Naïve Bayes for each of the sampling algorithms with respect to the number of labeled instances used. Fig. 2.5 and Fig. 2.7 are close-ups of leftmost part of the curves in Fig. 2.4 and Fig. 2.6, 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.

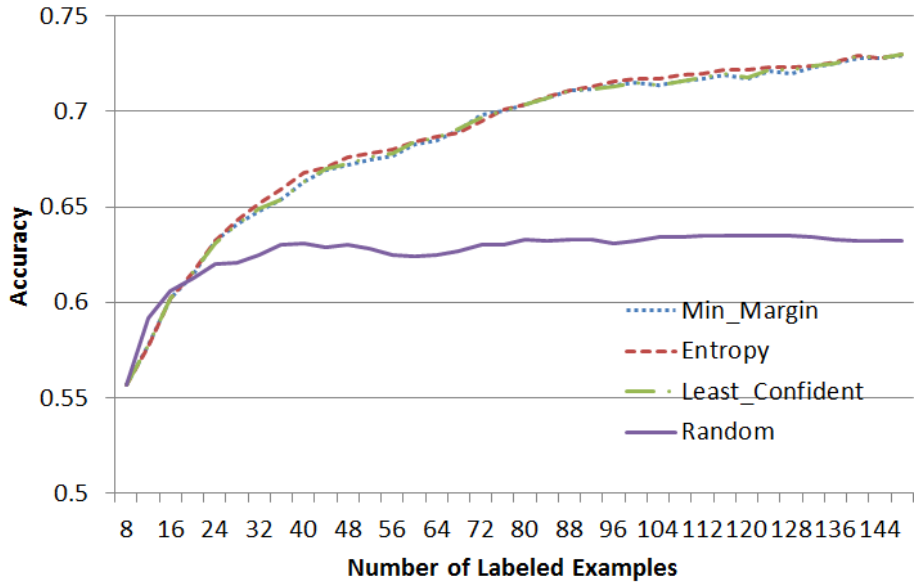


Fig. 2.5: Close-up view of Naïve Bayes accuracy

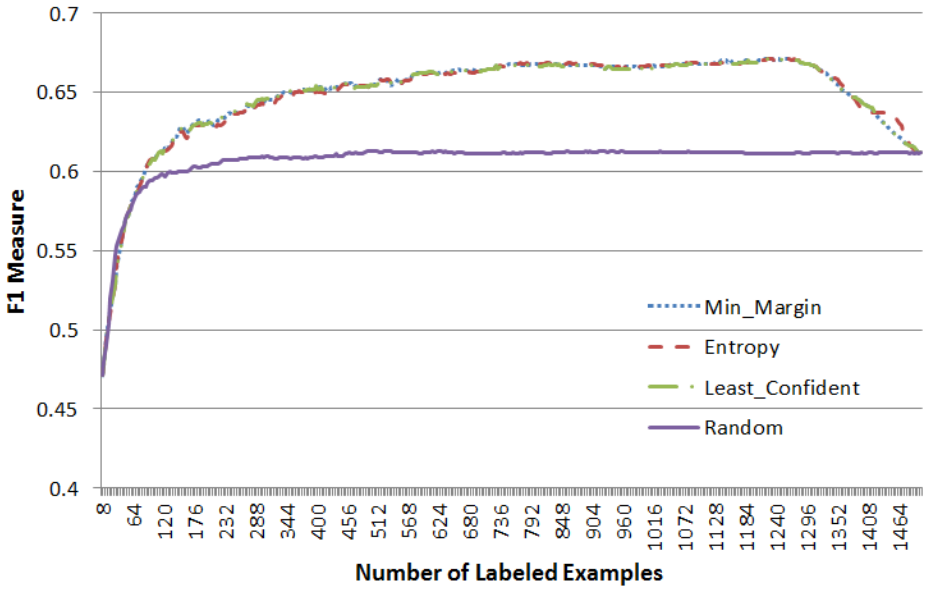


Fig. 2.6: Full simulation for Naïve Bayes F1

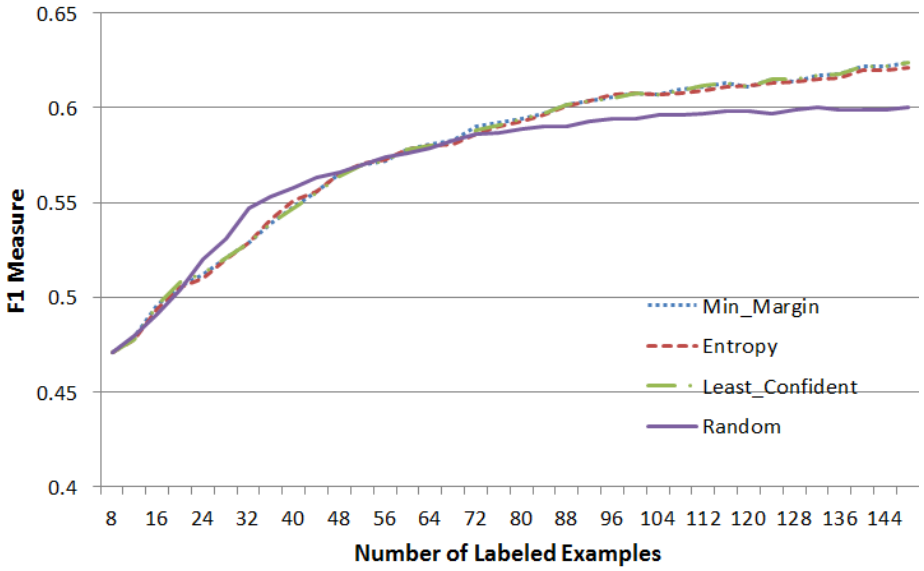


Fig. 2.7: Close-up view of Naïve Bayes F1

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 2.6.3, 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. Fig. 2.8 and Fig. 2.10 show the accuracy and F-measure for several sampling strategies as a function of the number of annotated examples. Fig. 2.9 and Fig. 2.11 are the close-up views for Fig. 2.8 and Fig. 2.10 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 Fig. 2.8), 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.

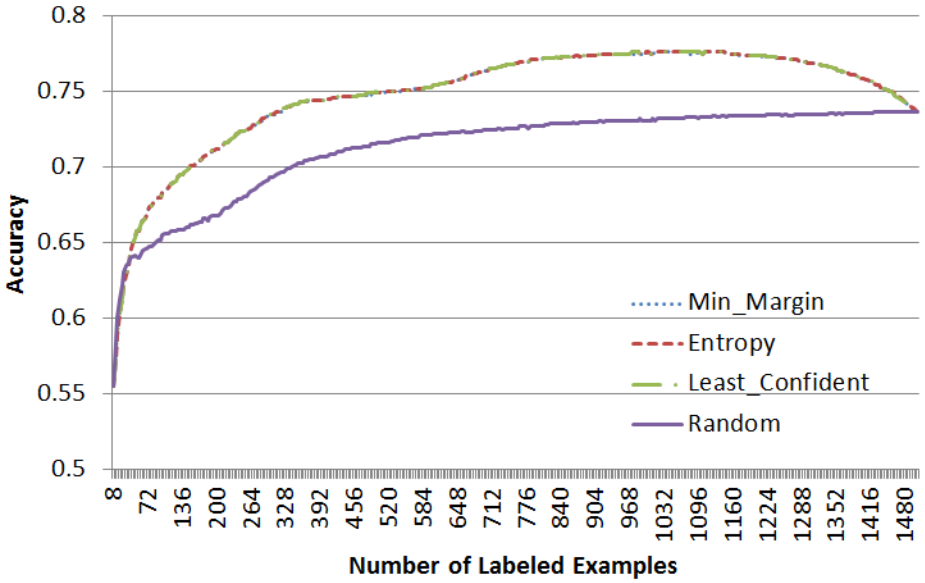


Fig. 2.8: Full simulation for committee accuracy

In the third experiment, we focused on the effect of 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.

To this end, we chose Naïve based active learning with entropy based

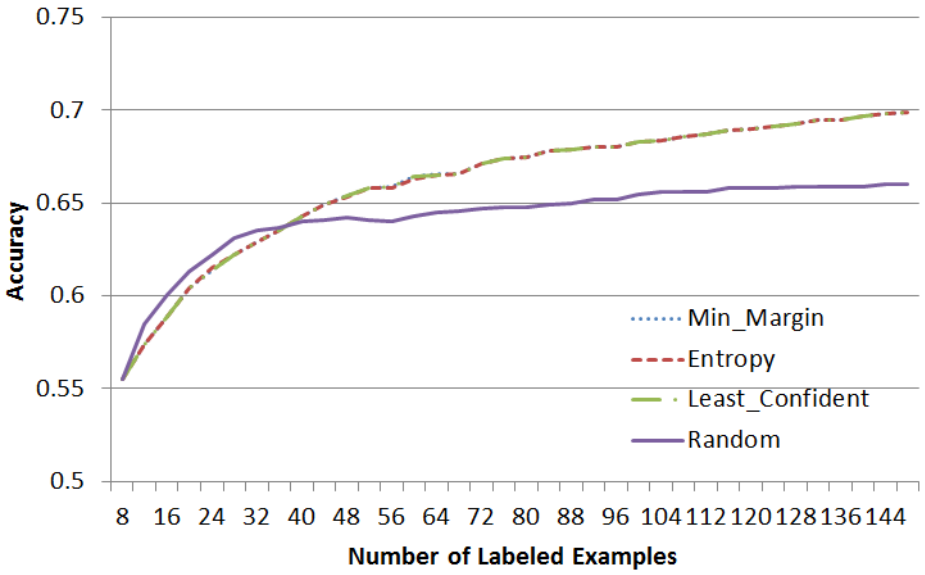


Fig. 2.9: Close-up view of committee accuracy

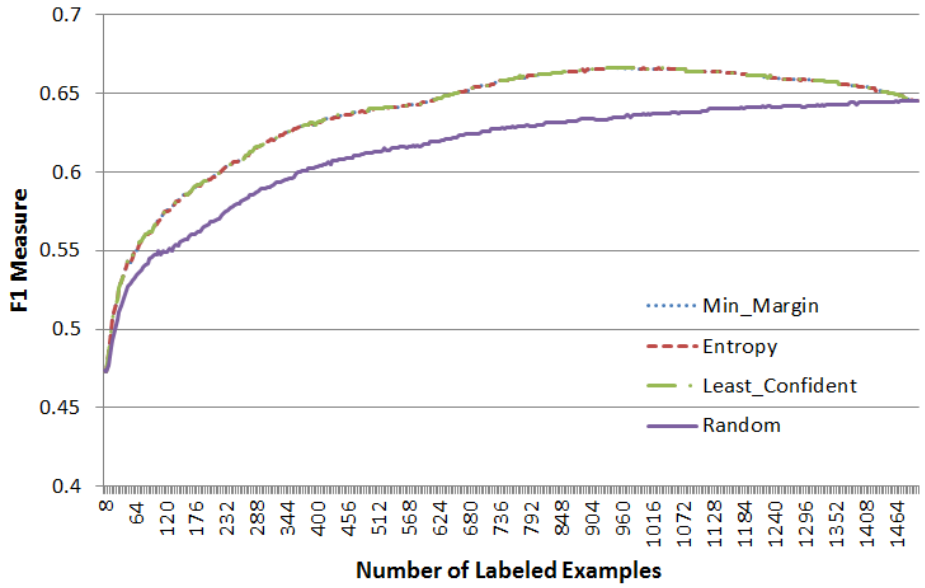


Fig. 2.10: Full simulation for committee F1

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 Fig. 2.12. 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.

In the last experiment, we varied initial seed size to see its effect of the initial

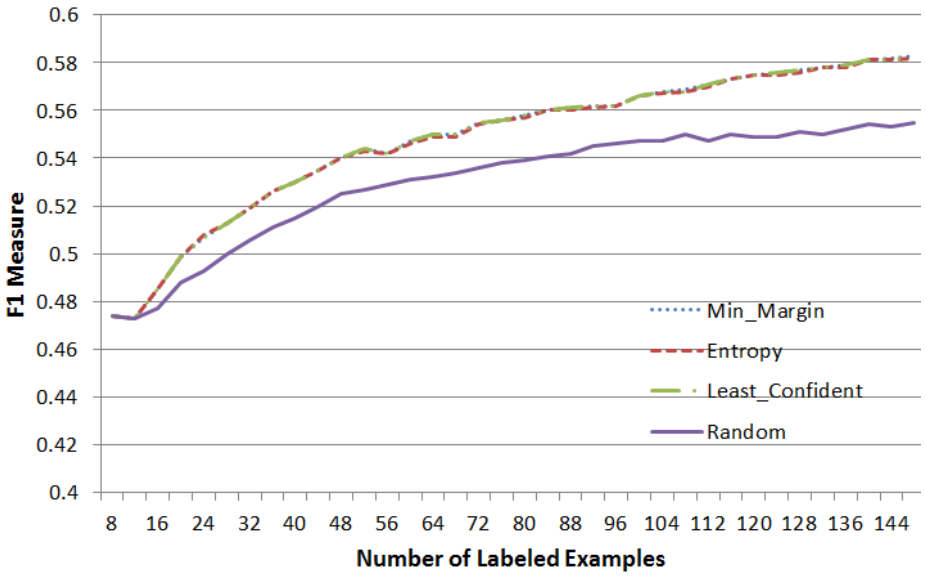


Fig. 2.11: Close-up view of committee F1

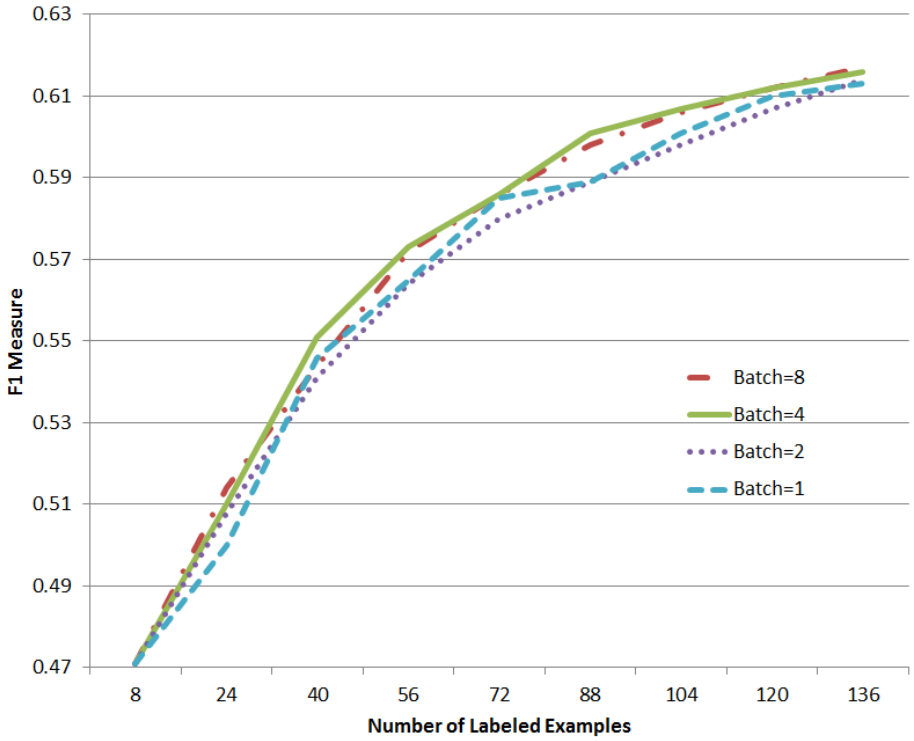


Fig. 2.12: Effect of batch size

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 Fig. 2.13 shows F1 measures of Entropy based sampling at

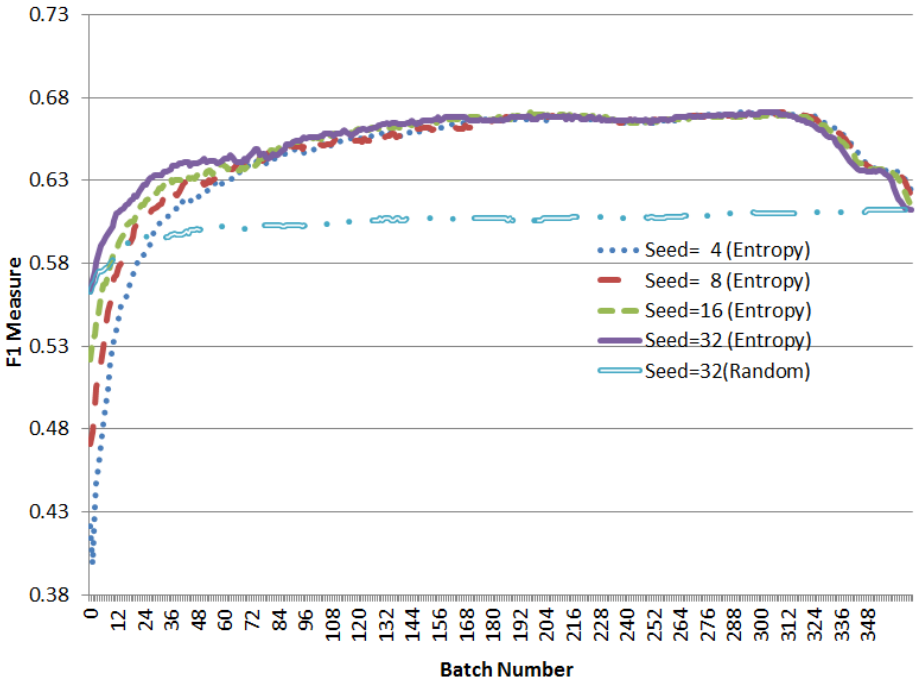


Fig. 2.13: Effect of seed data

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 Fig. 2.13. 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.

2.7 Discussions and Conclusions

This chapter discussed several issues related to automatically generating pedagogically good quality questions in the context of tutorial dialog. We started by presenting an approach that automatically mines cloze questions from recorded tutorial dialogues between actual students and a state-of-the-art ITS unlike most of the existing systems that rely on the content of an instructional text. We used the responses given by students to open-cloze questions to identify potential distractors for the gap-fill questions. For the questions that had less number of student

responses, we used WordNet to extract distractors similar to previous approaches proposed by others. We proposed different ranking functions to prioritize the list of potential distractor candidates including the one based on the frequency of students responses (votes). As such, this would be particularly useful for MOOCs and scalable ITSs where thousands of student solve the same problem. The proposed method can be applied to generate dynamic gap-fill questions and to make the assessment and dialog interactions more realistic.

Next, we presented a work that used active learning for training classifiers for judging the quality of automatically generated open-cloze questions. Experiments showed that active learning is very useful for creating cost-efficient methods for training question quality classifiers. 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.

Chapter 3

Anaphora Resolution

Anaphora resolution is one of the key problems to be addressed in order to understand students' responses. This chapter presents a machine learning approach to deal with this problem based on my published work (Niraula & Rus, 2014).

3.1 Introduction

The task of anaphora resolution is to identify the referent of a pronoun in dialogue and discourse. It is one of the important tasks in many NLP applications such as information extraction, automated essay grading, and summarization. It also plays a critical role in conversational ITSs as it can increase the accuracy of assessing students' knowledge level, i.e., mental model, based on their natural language inputs.

Students' natural language responses to tutors' questions are major sources of information about what a student knows. Incorrect assessment of student responses could lead to incorrect feedback provided by the system which, in turn, could frustrate students sometimes to the point of quitting using the system, an undesired effect. Because student responses often contain pronouns, the accuracy of the inferred student model is directly dependent on resolving anaphors in such student responses.

Consider the real student-tutor interaction below from a state-of-the-art ITS, DeepTutor:

PROBLEM: A mover pushes a desk with constant velocity V_0 across a carpeted floor. Suddenly, the mover stops pushing. What can you say about the motion of the desk after the mover stops pushing ? Explain why.
*STUDENT ANSWER: The desk will stop moving because **it** was only moving due to the applied force of the mover pushing on **it**. **It** does not have a constant velocity or*

Table 3.1: Use of pronouns in student responses

(a) Intra-turn :
TUTOR:What does Newton's second law say?
STUDENT:for every force, there is another equal force to counteract *it*
(b) Inter-turn immediate:
TUTOR:What can you say about the acceleration of the piano based on Newton's second law and the fact that the force of gravity acts on the piano?
STUDENT: *It* remains constant.
(c) Inter-turn history:
TUTOR: Since the ball's velocity is upward and its acceleration is downward, what is happening to the ball's velocity?
STUDENT: increasing
TUTOR: Can you please elaborate?
STUDENT: *it* is increasing

acceleration to keep ***it*** *going.*

The student answer in the example above has four pronouns, all referring to *desk*. To fully understand the student response these pronouns must be resolved. A pronoun resolution algorithm such as the one proposed here could help resolve the four pronouns. The need for such an algorithm is further emphasized by the fact that students' use of pronouns while conversing with a computer tutor is quite frequent (Niraula, Rus, & Stefanescu, 2013). The authors reported 5,881 pronouns in 25,945 student turns. Moreover, our analysis shows that about 22% of the total students turns contain at least one pronoun.

Three types of anaphora usage in students' answers can be identified in student-tutor interactions. They include *Intra-turn*, *Inter-turn intermediate* and *Inter-turn history* anaphora - see Table 3.1. In the case of *Intra-turn* anaphora, the referents are found within the student's current dialogue turn. In *Inter-turn intermediate* anaphora, the referents lie in the most recent tutor turn (Rus, Stefanescu, et al., 2014) and in *Inter-turn history* anaphora, the referents are located in earlier dialogue turns or even the problem description.

While anaphora resolution is a well-studied problem in written texts (Mitkov, 1999; Poesio & Kabadjov, 2004; Rahman & Ng, 2009) and dialogue (Poesio, Patel, & Di Eugenio, 2006; Stent & Bangalore, 2010; Strube & Müller, 2003), there are very limited works which address anaphora resolution in dialogue based ITSs which are more specific systems with different assumptions. Due to the peculiarities of tutorial dialogues, existing solutions for anaphora resolutions must be adapted to get optimal resolutions of anaphors in ITSs dialogues. To this end, we propose Deep Anaphora Resolution Engine++ (DARE++) for resolving pronouns in conversational ITSs. DARE++ is the improved version of DARE (Niraula, Rus, & Stefanescu, 2013), a previously developed heuristics-based anaphora resolution engine for dialogue based ITSs. DARE++ is one of the first machine learning techniques proposed for resolving pronouns in ITSs. It is guided by thousands of student-tutor interactions obtained from a state-of-the-art tutoring system, DeepTutor.

3.2 Related Works

The more general problem of finding coreferents, i.e., words and expressions referring to the same entity or event, is called coreference resolution. Anaphora is the special case of finding referents of pronouns. The literature on anaphora / coreference resolution for written texts is rich (Mitkov, Evans, & Orasan, 2002; Poesio & Kabadjov, 2004; Qiu, Kan, & Chua, 2004; Rahman & Ng, 2009; Versley et al., 2008). Similarly, considerable work on resolving pronouns in dialogue can be found in the literature (Ferguson, Allen, Galescu, Quinn, & Swift, 2009; Poesio et al., 2006; Stent & Bangalore, 2010; Soon, Ng, & Lim, 2001; Strube & Müller, 2003).

Methodologies for resolving pronouns in dialogue and discourse can be classified into knowledge-poor and classification approaches. Knowledge-poor approaches rely on hand-crafted rules or heuristics. A simple rule based approach proposed for ITSs and closest to this work is by Niraula, Rus, and Stefanescu

(2013). The authors learned simple rules from few annotated instances and applied them on top of an existing state-of-the-art coreference tool. The limitation of their approach is that learned rules using a few hundred observations is not sufficient for handling all the cases. Moreover, peculiar characteristics of the dialogue based ITSs are underutilized.

Classification approaches, on the other hand, work by means of models acquired through annotated corpora using machine learning algorithms. One such example is by Soon et al. (2001) who used a decision tree algorithm for coreference resolution. Strube and Müller (2003) proposed a machine learning approach to resolve pronouns in spoken dialogue. They also used decision tree to classify valid antecedent-pronoun pairs using their feature sets. Stent and Bangalore (2010) used logistic regression for mention-referent classification. Kernel based methods are also found in the literature to classify the pairs (Yang, Su, & Tan, 2006).

Anaphora resolution techniques proposed for English written texts need to be adapted when applied to texts in specific domains, genres (e.g., dialogue) and languages (other than English) as anaphora instances exhibit different characteristics than in professionally written texts such as newspaper articles. The technique proposed by Arregi et al. (2010) is such an example where authors adapted existing anaphora solutions in English to the Basque language. Similarly, Stent and Bangalore (2010) adapted solutions to resolve pronouns in a spoken dialogue system by adding spoken dialogue related features to existing solutions. Anaphora resolution in biomedical texts is another example of such adaptation (Gasperin & Briscoe, 2008).

ITSs have some commonalities with spoken dialogue systems in that both use dialogues in the interactions. It should be noted that we used data from ITSs that interact with students through typed dialogue, i.e., a chatroom-like type of interaction as opposed to spoken dialogue interaction. Furthermore, the dialogues

are in the context of science learning while spoken dialogue systems were studied mostly for common tasks such as airline ticket reservations. In both systems, antecedents corresponding to anaphors belong to current or previous utterances. However, there are differences too. First, in spoken dialogue systems, the majority of pronouns are personal and demonstrative pronouns (Strube & Müller, 2003). However, in tutorial dialogues, the pronouns are mostly *it*, *they*, *he* and *she* (Niraula, Rus, & Stefanescu, 2013). Second, referents can be VP-antecedents or NP-antecedents in spoken dialogue systems but almost all antecedents in ITSs are NP-antecedents.

Given the above peculiarities of tutorial dialogues compared to written texts and spoken dialogues, existing approaches to pronoun resolution should be adapted in order to maximize accuracy. To this end, we have proposed DARE++ that resolves anaphors in ITS dialogues using machine learning approaches.

3.3 Data

We extracted and annotated 1,000 pronoun instances from student-tutor interaction logs collected in an experiment involving high-school students interacting with the intelligent tutoring system DeepTutor in the domain of conceptual Physics. We described the details of the data set creation at Niraula, Rus, Banjade, et al. (2014). The data is freely available for public usage[1].

A typical collected instance is presented in Table 3.2. Each instance has a unique id (e.g., 3,624 in the example). The log files are records of the actual dialogue between the computer tutor and students. Student's current response is designated by A (student answer) and the corresponding utterance from the tutor, usually in the form of a guiding question from DeepTutor, is denoted by Q. Previous student responses are denoted with A1, A2, and so on, while previous DeepTutor turns are denoted with Q1, Q2, and so on. The goal is to resolve pronouns in A to

[1]http://language.memphis.edu/nobal/AR

Table 3.2: A typical instance for anaphora resolution

INSTANCE: 3624 PROBLEM: A stuntman must drop from a helicopter onto a target on the roof of a moving train. The plan is for the helicopter to hover over the train, matching the train's constant speed before the stuntman drops. Q2: Where should the helicopter be positioned relative to the target? Please begin by briefly answering the above question. After briefly answering the above question, please go on to explain your answer in as much detail as you can. A2: in front of the target due to wind resistance Q1: Let me try again. Which principle can be applied when the motion of an object is complex, for instance, it can be thought of as motion in two perpendicular dimensions? A1: decomposition Q: What can you say about the motion of the stuntman </np> after he jumps? A: it will be parabolic

their referent, which could be in the same student response A, the previous tutor turn Q, earlier in the dialogue history (and thus part of the common ground built by the two conversation partners), or even the current problem description.

Once the set of 1,000 instances was collected, we annotated the instances following a set of guidelines developed by linguistics experts (Niraula, Rus, Banjade, et al., 2014) which also borrowed some ideas from the guidelines used for annotating the data set used in the Message Understanding Conference (MUC-6 [2]). For annotation, we formed five pairs of annotators and trained them to annotate the instances. Each annotator in a pair annotated the same 100 instances independently, resolved their differences on the first 100 instances before repeating the annotation for another 100 instances. Average kappa statistic for the annotation was 0.84.

Once the annotated corpus was ready, we analyzed the annotated instances to first understand pronoun usage in our tutorial dialogues (see Table 3.3). A student answer can contain more than one pronoun and each pronoun may or may not have a referent (due to pleonastic pronouns, elipsis, etc.). About 78% of the

[2]http://www.cs.nyu.edu/cs/faculty/grishman/muc6.html

Table 3.3: Distribution of anaphors

Pronouns	**Count**	**Percentage %**
hasRef (e.g., it, he, she)	1003	78.11
first person personal pronouns	170	13.23
pleonastic	32	2.49
communication breakdown (Soft)	32	2.49
communication breakdown (Hard)	27	2.10
others	20	2.49

pronouns have referents, clearly demanding a method to resolve them. Students also used first person personal pronouns (e.g., I, we, and my) in their responses. About 13% of the pronouns are pleonastic. About 2.49% of pronouns need some form of inference to correctly identify their referents as the student answer does not precisely refer to an explicitly mentioned referent. We call such designate such case *communication breakdowns - soft*; (Niraula, Rus, Banjade, et al., 2014)). About 2.1% of pronouns' are found to be irrelevant to the context such that it is very hard to find their referents even by human experts (*communication breakdown - hard*' (Niraula, Rus, Banjade, et al., 2014)).

Table 3.4 shows the most used pronouns sorted by their frequency. The pronouns *it, they and its* are the three most frequent pronouns and account for more than 70% of pronoun usage. Since *it* can be pleonastic, identifying and resolving this pronoun is particularly challenging.

We further generated statistics about the locations of the referents corresponding to the students' pronouns and presented the top locations in Table 3.5. More than 50% of the pronouns refer entities in Q (the immediate tutor question), about 30% of the pronouns have their referents in A (i.e., in the student answer as the pronoun to be resolved), and about 11% of the referents are found in the problem descriptions (Ps). Very few pronouns refer to the entities in the previous tutor questions in the dialogue history (Q_i).

Table 3.4: Most common pronouns

Pronoun	**Count**	**Percentage(%)**
it	658	53.47
they	153	11.94
its	120	9.37
i	61	4.76
you	55	4.29
her	36	2.81
she	34	2.65
them	21	1.63
he	19	1.48
their	18	1.40
his	17	1.33

3.4 Methodology

Machine learning based methods are among the most popular approaches to the problem of coreference resolution (Stent & Bangalore, 2010). The standard coreference pipeline for such methods include identification of *mentions* which are co-referring expressions, extraction of *features* describing these mentions, determining *mention-pair* candidates which are pairs of mentions that corefer, and *clustering* mention-pairs in order to identify mentions that form a chain, i.e., refer to the same entity.

Table 3.5: Top five locations for antecedents

Location	**Count**	**Percentage(%)**
Q	577	53.22
A	342	31.54
P	125	11.53
Q_1	28	2.6
Q_2	5	0.46

We adopted this coreference pipeline with some modifications. First of all, we do not generate all mention-pairs as our objective is not to generate the complete coreference chain rather just resolve the pronouns in the students answer to the corresponding entity, typically the most recent non-anaphoric reference of the

entity. That is, we are interested in finding the referents (if any) of only pronouns that appear in a student answer but not necessarily finding chains of pronouns or other types of referents to the same entity. This is sufficient for our goal of best understanding the current student answer. This simplification significantly reduces the search space of mention-pairs. Additionally, we do not need to cluster the mentions as we need only one referent of a pronoun. Thus, our model generates a limited number of mention-pairs and classifies them as either P(ositive) which means the two mentions (typically a noun and a pronoun mention) corefer or N (egative), otherwise. We present next the major phases in our anaphora pipeline.

3.4.1 Generation of Mention-pairs

Our mention-pair construction algorithm works as follows. We use a parser to parse the problem text and tutor-turns and extract noun and noun phrases (we do not consider previous student turns for mention candidates as pronouns in student answers almost never refer to something in a previous student answer/turn). Next, we parse student's answer (i.e., A) and identify pronouns to be resolved. These pronouns are then paired with nouns to get mention-pairs. We exemplify this process for the instance shown in Table 3.2. We parse the sentences in PROBLEM, Q2, Q1, Q, and A and get the following mention-pairs: (*stuntman,it*), (*helicopter,it*), (*target,it*), (*room,it*), (*train,it*), (*principle,it*), (*motion,it*), etc.

3.4.2 Feature Selection

In order to use machine learning techniques to automatically induce a classifier, we need to devise a set of features that are useful for classifying the mention-pairs as P or N. This is a crucial step as the accuracy of the induced classifiers relies significantly on these features. We used lexical, syntactic, semantic, and dialog related features which are listed in Table 3.6.

Lexical Features: Lexical features include lengths of A, Q, A_1, Q_1, A_2, and Q_2, pronoun (P)'s position in A (i.e., the token index), total number of pronouns in

Table 3.6: List of features (P = pronoun, C = a referent candidate)

Type	Features
Lexical	lengths (of A, Q, A_1, Q_1, A_2, and Q_2)(1-6), P's token position in A (7) no. of Ps in A(8), % of tokens before & after C (10-11), is C in A ? (12) has WH-Word in A(13), has negation word in A? (14), question type (15)
Syntactic	dependency relation counts (governer & total) of P (19-20) and C (21-22), present/absent 135 dependency relations (24-158)
Semantic	gender agrees ? (16), number agrees ? (17), person of P (18), is C a proper noun ?(23)
Dialogue	location of C in dialogue stack (9)

student's answer, percentage of tokens before and after a referent candidate (C). We also have boolean features to check whether the candidate referent C is in student's answer A, whether student's answer contains any WH-words and simple negative cue words. We used lists of WH-words and negative cue words, respectively, for this purpose. Type of question is determined by checking first token in Q in this list: (what:1, when:2, where:3, which:4, who:5, whom:6, whose:7, how:8, none of above:-1).

Syntactic Features: To capture the grammatical functions of antecedent candidates, we counted the number of dependency relations and the number of relations with the candidate being a head word (governor). We also computed these features for pronouns. Moreover, we used binary features for 135 dependency relations each indicating true when the referent candidate is either its governor or dependent.

Semantic Features: We created a dictionary to get the gender of pronouns and of the characters (e.g., John) used in the problem descriptions. Values of *gender agrees* feature can be 1 (matched), 0 (not matched) and 2 (not available). For the *number*(s/p/na), we use simple rules using POS tags. For example, if a noun's POS is NN or NNP, we considered that noun a singular whereas if the POS is NNS or NNPS we consider it as a plural(p). Similarly, a noun is deemed a proper noun if its

first character is capitalized (which can also be detected through the NNP or NNPS tags).

Dialogue Features: We used one dialogue feature: *the location of candidate* referent which takes value from 0 to 9 (A:0, Q:1, A_1:2, Q_1:3, A_2:4, Q_2:5, A_3:6, Q_3:7, problem description:8, none of above:9).

3.4.3 Generation of Training Examples

The machine learning algorithms we experimented with require both positive and negative instances in order to learn the target function, which in our case is a classification function. We generated positive (P) and negative (N) examples of mention-pairs using the annotated data set. Note that an example (training or testing) is a vector containing values corresponding to the feature set. Positive examples are easy to generate as pronouns and their correct referents are marked in the annotated instances. For example, for the instance in Table 3.2, we generate the following positive mention-pair (*motion,it*).

To generate negative examples, we follow an approach similar to (Soon et al., 2001). Following this approach, we generate negative examples by using *(entity, pronoun)* pairs where *entity* refers to any noun between the pronoun and its annotated referent. To achieve this, we start going backwards from the pronoun to be resolved and scan for nouns until we reach its correct referent. We form *(noun, pronoun)* pairs for every identified noun in this span of dialogue. All the pairs except *(correct-referent,pronoun)* are used to generate negative examples. As an example, we generate the following negative instance of a mention-pair from the annotated instance in Table 3.2: (*stuntman,it*). This mention pair is negative because *stuntman* is between the pronoun *"it"* and its referent *"motion"*. If we had other entities like *stuntman* in between *"it"* and *"motion"*, we would have generated other negative examples as well.

Table 3.7: Performance comparison

Method	Acc.	Pre.	Rec.	Fm.	Kappa	RME
Baseline (Niraula, Rus, & Stefanescu, 2013)	39.10	38.03	53.96	44.26	-	-
Naive Bayes	82.33	66.9	78.11	72.11	0.59	0.35
SVM	87.78	84.06	71.83	77.47	0.69	0.34
Logistic Regression	88.06	81.24	76.85	79.00	0.70	0.29
Decision Trees (J48)	**93.54**	**89.07**	**88.79**	**88.93**	**0.84**	**0.24**
Multilayer Perceptron	88.82	86.96	72.67	79.17	0.71	0.31

3.4.4 Resolution of Mention-Pairs

To automatically learn how to classify a mention-pair as P or N, we used a number of classifiers which were trained using the positive and negative examples described in Section 3.4.3. Once induced, the classifier can be used to classify future instances as either P or N. For instance, the referent of a new pronoun would correspond to the referent in the mention-pair classified as P.

3.5 Experiment Setup and Results

We used the previously mentioned technique to extract the positive and negative examples from the annotated corpus. In total, we obtained 955 positive and 2,312 negative examples. Although our corpus has 1,000 annotated instances, the positive examples are less because not all pronouns in the annotation corpus has a referent (e.g., pleonastic pronouns, etc.). We considered the examples corresponding to pronouns without any referents as negative examples as we want our classifier to learn to reject such pairs in the future.

We used ten-fold cross-validation on the 3,267 examples for a number of classifiers as done by Arregi et al. (2010). For comparison purpose, we used the DARE system (Niraula, Rus, & Stefanescu, 2013) as our baseline. Results are reported in terms of precision, recall, accuracy, F-measure and kappa statistic.

Table 3.7 shows the results for the baseline, and the best results obtained for DARE++ using Naive Bayes, Support Vector Machine (SVM), Logistic Regression, Decision trees and Multilayer perceptron classifiers. The results reported were obtained after tuning various parameters of these machine learning algorithms.

Note that all the classifiers have large performance gains over the baseline (Niraula, Rus, & Stefanescu, 2013) in terms of accuracy, precision, recall and F-measure scores. It is found that poor performance of the baseline system is due to its fairly simple assumptions about the referents' locations which do not cover all the cases. For instance, one simple rule in the baseline algorithm stipulates that referents of pronouns that occur in the middle of a student answer are located in the same student answer (Niraula, Rus, & Stefanescu, 2013). While this seems right for some cases, it is often not true.

Among all classifiers, Logistic Regression, Decision Tree (J48) and Multilayer Perceptron are the best performing classifiers in terms of F-measure, Kappa-statistic and the root mean squared error (RME). These classifiers have F-measures over 79%. Decision Tree using J48 has the highest accuracy, precision, F-measure and Kappa statistics and the lowest root mean squared error.

For tutorial dialogues, false positives are very important because declaring a noun as a referent of a pronoun, when it was actually not, leads to a different interpretation of the student's response. On the other hand, false negatives are less sensitive than false positives as they do not add wrong information during the interpretation process (e.g., suggesting *a pronoun does not have a referent when it had one* is not as severe as suggesting *an pronoun has a referent when it didn't have one*). Thus, we paid attention to the false positive counts of the classifiers. We found that the best performing classifiers also have lower false positive counts, satisfying the conditions for tutorial dialogues.

3.5.1 Feature Analysis

We experimented with adding unigrams, bigrams, and trigrams features for the tokens in A, Q and Q_i and their part-of-speeches as done by Stent and Bangalore (2010) for spoken dialogues. However, the performance didn't improve. Thus, the set of features presented in Table 3.6 is the best for tutorial dialogues.

We further studied the features in order to understand which ones are the most informative in tutorial dialogues. We used information gain and gain ratio to rank the features. The most informative features turned out to be: *the location of referent*, *prep_about* (dependency relation), *% of tokens after candidate*, *number agrees?*, *gender agrees?*, *det* (dependency relation), *governor relation counts for candidate*, *is candidate a proper noun*, *person of pronoun*, *prep_of* (dependency relation).

It is not surprising to see that the gender, number, and person features are crucial while determining the referents of pronouns in general. Interestingly, *the location of referent* is one of the most informative features for anaphora resolution in tutorial dialogues, which is different compared to the role of this feature in anaphora resolution for written texts. As suggested by the Table 3.5, more than 80% of the antecedents are located in Q and A alone. *governor relation counts for candidate* is another informative feature in tutorial dialogues. This is the case because words with many governor relations are more likely to be the focus of the tutor question which is typically referred by students in their answers. The dependency relations such as *prep_about* and *prep_of* are found to be other useful features for tutorial dialogues. The tutors typically ask students the following type of questions: *What can you say about XX of the YY ?* Student may reply with: *It equals ZZ.* In such examples, the pronoun *it* in the student answer refers to XX in tutor's question which is involved in a *prep_about* relation. Due to the relative high frequency of such tutor question - student answer pattern the *prep_about* relation becomes salient.

3.6 Discussions and Conclusions

In this chapter, we presented a solution to the problem of pronoun resolution in tutorial dialogues obtained from dialogue-based ITSs. Although pronoun resolution for written texts and spoken dialogues is well studied, it is not explored

much for tutorial dialogues. Our experiments show that DARE++ can achieve a F-measure of 88.93%, showing its robustness in resolving pronouns.

Although the performance of DARE++ is impressive, it can be improved further. Demonstrative pronouns, ellipsis, soft, and hard communication breakdowns (see Table 3.3) are the major factors limiting its performance. Next important factor is having pronouns without antecedents (e.g., pleonastic pronoun). In addition, we have not considered *cataphora* currently. They are less frequent in tutorial dialogues but should be handled to make the system more robust. Finally, we would like to explore other models that use a reduced set of features based on the feature analysis we presented here or future feature analyses.

Chapter 4

Assessment of Student Responses

In this chapter, we seek to find automatic and effective methods for assessing students' responses. Such methods play a crucial role in automatic dialog generation since the type of feedback to students depends on the effectiveness of the methods. Since the students' responses and corresponding answers are typically short sentences, the concentration should be on exploring the efficient methods for finding semantic similarity between two short texts. Nevertheless, assessing how semantically similar two short texts is very challenging.

4.1 Semantic Textual Similarity

Semantic Textual Similarity (STS) is the task of measuring the degree of semantic equivalence for a given pair of texts. The problem is a central topic in Natural Language Processing (NLP) as it plays a crucial role in many NLP applications such as providing evidence for the correctness of answers in Question Answering (Ibrahim, Katz, & Lin, 2003), increasing diversity of generated text in Natural Language Generation (Iordanskaja, Kittredge, & Polguere, 1991), assessing the correctness of student responses in Intelligent Tutoring Systems (Graesser et al., 2005), and identifying duplicate bug reports in Software Testing (Rus, Nan, Shiva, & Chen, 2009). More specifically, the task of semantic similarity involves making a judgment with respect to how semantically similar two texts are. The judgment can be quantitative, e.g., a normalized score, or qualitative, e.g., one text is (or not) a paraphrase of the other.

For instance, in a conversational Intelligent Tutoring System, it is important to understand students' natural language inputs in order to assess their level of understanding of the target topic to be learned and, consequently, provide appropriate feedback (Rus et al., 2013). One frequently used approach to address this student input assessment problem is to compute how similar the student

response is to a benchmark response such as an expert-articulated response (Graesser et al., 2005; Rus & Graesser, 2006). That is, student response assessment task is modeled as a text-to-text similarity problem. Here is an example of a real student response from an ITS and corresponding benchmark answer authored by an expert:

Student Response: An object that has a zero force acting on it will have zero acceleration.

Expert Answer: If an object moves with a constant velocity, the net force on the object is zero.

The student response above is deemed correct as it is semantically similar to the expert answer. A student response is deemed incorrect if it is not similar to the expert response. It should be noted that this type of binary modeling, which we adopt in this work, has been extensively used in previously proposed semantic similarity tasks such as the Recognizing Textual Entailment task (Dagan, Glickman, & Magnini, 2006), the paraphrase identification task (Dolan, Quirk, & Brockett, 2004), or the student input assessment task (McCarthy & McNamara, 2008; Rus & Graesser, 2006). More nuanced categorizations are possible, e.g., a student response can be partially correct.

4.1.1 Relatedness and Similarity Measures

Basically, two types of measures are used to find the semantic relations between texts: *similarity* and *relatedness* measures. Although they are related, there are subtle differences between them. For instance, *chicken* and *egg* are related as they often appear together, but they are not similar (living vs non-living). Thus, similarity focused measures quantify the meaning shared by two words and relatedness focused methods quantify the associations between the words.

4.1.2 Vector Algebra for Semantic Similarity

When a word or a text is represented as a vector in semantic space, vector algebra can be applied to compute semantic similarity among them. For example, to compute the semantic similarity between word W_i and word W_j, we use the cosine similarity between word vectors as :

$$Sim(W_i, W_j) = \frac{\sum_{n=1}^{K} V_i[n] * V_j[n]}{|V_i| * |V_j|}$$

where, V_i and V_j are the vectors corresponding to word W_i and W_j respectively, and K is the dimension of vector V_i (or V_j).

Another advantage of representing a word as a vector is that we can compute the semantic representation of a longer text (e.g., a sentence) by simply summing up the individual word vectors of the text. That is equivalent to finding the resultant vector of individual vectors. Once the representation of a text is obtained, we can again compute the cosine similarity to compute its similarity with a given word or text.

4.1.3 Sentence-level Semantic Similarity using Word-to-Word Similarity

The task of semantic similarity can be formulated at different levels of granularity, ranging from word-to-word similarity, to sentence-to-sentence similarity, to document-to-document similarity, or a combination of these, such as word-to-sentence or sentence-to-document similarity.

Two categories of measures are popular in the literature to compute sentence-level semantic similarity: those that compute similarity more globally and those that rely on word-to-word similarity measures. The global approach derives a semantic representation of entire text/sentence at once, without composing one from word representations or from word-to-word semantic similarity measures. The representations for sentences are then used to compute the semantic similarity among them. For instance, semantic similarity between two sentences can be

computed by computing the cosine similarity between the corresponding semantic representation vectors.

The later approach computes sentence-to-sentence semantic similarity by exploiting word-to-word similarity measures. One simple approach is first to generate sentence representation by summing up the individual word representations and then compute the cosine similarity between two sentences by using the representations. Other approach computes semantic similarity by combining the word-to-word similarities using some greedy or optimal matching method.

Greedy Matching : In the greedy approach words from one sentence (usually the shorter sentence) are greedily matched, one by one, starting from the beginning of the sentence, with the most similar word from the other sentence. The matching between two words is quantified by various word-to-word similarity measures. In case of duplicates, the order of the words in the two sentences is important such that the first occurrence is matched with the first occurrence and so on. To be consistent across all methods presented here and for fairness of comparison across these methods, we require that words must be part of at most one pair. It should be noted that others, e.g., Corley and Mihalcea (2005), did not impose such a requirement and therefore, some words could be selected to be part of more than one pair.

The greedy method has the advantage, over the other methods, of being simple and fast, while also effectively using the natural order of words within the sentence. The obvious drawback of the greedy method is that it does not aim for a globally maximum similarity score. The optimal method described next solves this issue.

Optimal Matching : The optimal method aims at finding the best overall word-to-word match. This is a well-known combinatorial optimization problem called the assignment problem. The assignment problem consists of finding a maximum weight matching in a weighted bipartite graph. Its instantiation to a job

assignment context is most famous. Given a complete bipartite graph, $G = (T_1, T_2, E)$, with n worker vertices (T_1), n job vertices (T_2), and each edge $e_{s \in T_1, t \in T_2} \in E$ having a non-negative weight $w(t_1, t_2)$ indicating how qualified a worker is for a certain job, the task is to find a matching M from workers (T_1) to jobs (T_2) with maximum weight. In case of different numbers of workers or jobs, dummy vertices could be used.

The assignment problem can be formulated as finding a permutation for which $S_{OPT} = \sum_{i=1}^{n} w(t_{1i}, t_{2\pi(i)})$ is maximum. Such an assignment is called optimum assignment. An algorithm, the Kuhn-Munkres method, has been proposed that can find a solution in polynomial time (Kuhn, 1955).

4.2 Literature Review

We will review two major research areas that are most related to our work: research on word-to-word similarity measures and research on text-to-text similarity measures with a focus on sentence level similarity.

The literature for computing word-to-word similarity and relatedness is very rich. Broadly, these methods can be categorized into three groups depending on the type of resources they use: Knowledge-based, Corpus-based and Web-based. Knowledge-based methods rely on some form of ontology. WordNet (Miller, 1995) is a well-known knowledge source that has been widely used to compute the semantic similarity and relatedness between words. It is a large lexical database of English consisting of nouns, verbs, adjectives and adverbs that are grouped into concepts i.e., synsets (synonym sets). The concepts are then linked through lexico-semantic relations such as hypernymy (is-a type of relation). The graph of lexicons has been exploited in different ways resulting in several similarity measures (Banerjee & Pedersen, 2003; Hirst & St-Onge, 1998; Lin, 1998; Wu & Palmer, 1994).

Corpus-based measures compute word similarity / relatedness scores based on the words' representations obtained from a given corpus. LDA, LSA and Explicit

Semantic Analysis (ESA) (Gabrilovich & Markovitch, 2007) are some of the most popular approaches for inferring word representations based on which a number of approaches have been devised (Rus, Lintean, Banjade, Niraula, & Stefanescu, 2013). Most recently, neural models have been proposed to derive word representations from a corpus (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013; Turian, Ratinov, & Bengio, 2010). These measures have diverse assumptions and range from algebraic to probabilistic methods. Since we are going to combine these methods, we will give a more detailed account of these approaches in the next chapter.

The third category of measures use the Web as a source of information. Some people consider this method as a corpus-based method by considering the Web as a corpus. Measures in this category rely on web-search results such as page count statistics and text snippets to compute the similarity of words. In other words, these methods use search engines as proxies to gather word co-occurrence statistics. The major advantage is the sheer size of the documents that commercial search engines use which supposedly makes the co-occurrence statistics more reliable. For example, if two words W_1 and W_2 co-occur within the same web documents then a web search query such as W_1 AND W_2 will return many documents. The PMI-IR measure used by Turney (2002) and Bollegala, Matsuo, and Ishizuka (2007) are the examples in this category. PMI-IR is an unsupervised measure proposed by Turney (2002). The core concept of PMI-based techniques is that the similarity between words can be captured by using their statistical dependence. Web-based approaches are preferable over ontology-based approaches, such as the WordNet-based approaches, especially when the semantic similarity between words can change over time and across domains. However, offline similarity computation can be a challenge with such approaches.

4.2.1 LSA-based Semantic Similarity

LSA is a fully automated method that computes semantic vector representations for words from a given corpus (Landauer et al., 2013). It starts with a term-document matrix that represents the distribution of words in documents and the distribution of documents over the words. The word vectors (as well as the document vectors) in the original term-document matrix are mapped using the mathematical procedure of Singular Value Decomposition into a reduced dimensionality space. The dimension of the space is typically from 300 to 500. Words are represented as vectors in this LSA semantic space whose dimensions form latent semantic concepts. Documents are also represented as vectors in the reduced space. Similarity of individual words and texts are then computed based on vector algebra.

Lintean, Moldovan, Rus, and McNamara (2010) looked at the role of LSA in solving the paraphrase identification task. They used LSA as a way to compute semantic similarity in two different ways. First, they used LSA to compute a word-to-word similarity measure which they combined with a greedy-matching method to obtain a sentence level similarity score. For instance, each word in one sentence was greedily paired with one word in the other sentence. An average of these word-to-word similarities was then assigned as the semantic similarity score of the two sentences. Second, LSA was used to directly compute the similarity of the two sentences by applying the cosine (normalized dot product) of the corresponding LSA vectors of the two sentences. The LSA vector of a sentence was computed by simply adding individual word vectors corresponding to all the words in the sentence. They have not compared their results with any other unsupervised method such as LDA.

4.2.2 WordNet-based Semantic Similarity

As mentioned before, WordNet has been used to compute semantic orientation between two concepts. The semantic orientation can be semantic similarity or semantic relatedness. The former measures the similarity scores based solely on the *is-a* hierarchy of the synsets containing the concepts. A popular measure of this category is the LIN measure (Lin, 1998) which computes the semantic similarity between concept X and concept Y by using the information content of the least common sumsumer of X and Y. Specifically, it computes the *commonality* and *differences* between X and Y using information content. It then defines the similarity between X and Y as the ratio of *commonality* over *differences*.

Since semantic similarities use *is-a* relations, they are defined only for concepts belonging to a hierarchy e.g., for nouns. However, two concepts which are not connected with *is-a* relations can also be related. For example, *hand* and *body* are related through *a-part-of* relation. Such semantic orientation can be captured through semantic relatedness which is, therefore, not as strict as semantic similarity. LESK measure is an example of this category (Banerjee & Pedersen, 2003). This measure computes semantic relatedness based on the number of overlapped words in the word senses and glosses of the concepts in WordNet.

4.3 Short Text Semantic Similarity using LDA

Latent Dirichlet Allocation (LDA) belongs to the broader category of methods called topic models (Blei, Ng, & Jordan, 2003). Topic models are based on the assumption that a relatively small set of latent topics underlie natural language texts. The topics are groups of semantically related words. A word ranks differently in multiple topics. If one interprets each topic as being a concept then LDA directly models polysemy which LSA does not. By contrast, each word in LSA has a unique vector representation. That is, multiple senses of the same word are mapped to the same representation in the reduced LSA space. In fact LDA was proposed to

address several limitations of Probabilistic Latent Semantic Indexing (pLSI) model (Hofmann, 1999) and LSA (Blei et al., 2003). This theoretical advantage of LDA over LSA, when it comes to modeling word meanings, motivates us to identify which one is better at tasks in which word meanings play a role such as sentence-level text-to-text similarity.

4.3.1 Latent Dirichlet Allocation

LDA is a generative probabilistic model for collections of discrete items, i.e., words in our case. The only observables in an LDA model are the words in the documents. All else are latent variables. LDA derives the parameters of the latent variables using the observed words in the corpus. We say that LDA captures significant intra-document statistical structure via mixing distributions.

We will use the notation as in Blei et al. (2003) to explain the basic LDA model. A word, denoted w, is a discrete unit entry in a vocabulary V whose elements are indexed $\{1,...,V\}$. A document is a sequence of N words denoted $\mathbf{w} =< w_1, w_2, ..., w_N >$, where w_i is the i_{th} word in the document. A corpus D is a collection of documents $D = \{\mathbf{w}_1,\mathbf{w}_2, ...,\mathbf{w}_M\}$.

Documents are regarded as random mixtures of topics and a topic is a distribution over words in the vocabulary. LDA follows the following generative process for a document $\mathbf{w}$.

1. Choose a topic distribution $\theta \backsim Dir(\alpha)$; the dimensionality k (number of topics) of the Dirichlet distribution is given;

2. For each of the N words w_i in $\mathbf{w}$:

    i. Select a topic z_i based on θ

    ii. Choose a word w_i using $p(w_i|z_i, \beta)$

LDA has two Dirichlet priors: α for document-topic distributions and β for topic-word distributions. These two priors, α and β, are also known as

hyper-parameters for the document-topic and topic-word Dirichlet distributions. Although they can be vector valued, many LDA implementations use α and β as scalars to simplify and get symmetric Dirichlet priors. Furthermore, most LDA users choose symmetric Dirichlet priors using some heuristics. One such heuristics is mentioned by Griffiths and Steyvers (2004): although the values of these priors depend on vocabulary size and the number of topics, setting $\alpha = 50/k$ and $\beta = 0.01$ worked well for many different text collections. We followed this recommendation in our work.

LDA estimation includes learning the various distributions, e.g., the topic distributions over words. Estimation of the LDA parameters directly and exactly maximizing the likelihood of the whole data collection is intractable. Approximate estimation methods are used to solve the problem. Three popular methods are reported in the literature: variational methods (Blei et al., 2003), expectation propagation (Griffiths & Steyvers, 2004), and Gibbs sampling (Griffiths & Steyvers, 2004). We used an implementation based on Gibbs sampling (Phan, Nguyen, & Horiguchi, 2008).

Number of Topics

The standard LDA model requires the specification of the number of latent topics in advance. That is, the number of topics is set by the user. Choosing the right number of topics is important as they determine the quality of the LDA model. Choosing the right value for the number of topics is more art than science.

Nonparametric Bayesian models such as an Hierarchical Dirichlet process were also proposed to automatically estimate the number of topics (Teh, Jordan, Beal, & Blei, 2006). The nonparametric models are not computationally efficient though (Wallach, Mimno, & McCallum, 2009).

4.3.2 LDA-based Similarity Measures

LDA itself was occasionally used for computing the semantic similarity of texts. The closest use of LDA for a semantic similarity task was by Celikyilmaz, Hakkani-Tur, and Tur (2010) for ranking candidate answers to questions in Question Answering (QA). Given a question, they ranked candidate answers based on how similar these answers were to the target question. That is, for each question-answer pair they generated an LDA model which they then used to compute a degree of similarity (DES) that consists of the product of two measures: sim_1 and sim_2. sim_1 captures the word-level similarities of the topics present in an answer and the question. sim_2 measures the similarities between the topic distributions in an answer and the question. The LDA model was generated based solely on each question and its candidate answers. As opposed to our task, in which we compute the similarity between two sentences, the candidate answers by Celikyilmaz et al. (2010) are longer, consisting of more than one sentence. This particular difference is important when it comes to computing semantic similarity based on LDA as the shorter the texts the sparser the distributions, in particular the distribution over topics, based on which the similarity is computed. We will elaborate on this major point later.

As we already mentioned, LDA is a probabilistic generative model in which documents are viewed as distributions over a set of topics and each word in a document is generated based on a distribution over words that is specific to each topic. Therefore, two types of semantic similarity measures can be computed: using distributions over words and using distribution over topics.

The first semantic similarity measure, between two words, would then be defined as a dot-product between the corresponding vectors representing the contributions of each word to a topic, which could be generated based on the distributions over words for each topic. The basic idea here is that the more two

words contribute to same topics, the more similar they must be. It should be noted that the contributions of each word to the topics do not constitute a distribution, i.e., the sum of contributions does not add up to 1. Assuming the number of topics T, a simple word-to-word measure is defined by the formula in Equation 4.1 where we denote by ϕ distributions over words for a topic t. We normalize the score so that the similarity score will be between 0 and 1.

$$LDA - w2w(w, v) = \sum_{i=1}^{T} \phi_t(w)\ \phi_t(v) \tag{4.1}$$

More global similarity measures, between two texts as opposed to two words, could be defined in several ways. Because a document is a distribution over topics, the similarity of two texts needs to be computed in terms of similarity of distributions. The Kullback-Leibler (KL) divergence defines a distance, or how dissimilar, two distributions p and q are as in the formula below.

$$KL(p, q) = \sum_{i=1}^{T} p_i \log \frac{p_i}{q_i} \tag{4.2}$$

If we replace p with θ_c (document c's distribution over topics) and q with θ_d (document d's distribution over topics) we obtain the KL distance between two documents (documents c and d in our example).

The KL distance has two major problems. In case q_i is zero KL is not defined. Furthermore, KL is not symmetric which does not fit well with semantic similarity measures which in general are symmetric. That is, if text A is a paraphrase of text B that it is safe to say that text B is a paraphrase of text A. The

Information Radius (IR) measure solves these problems by considering the average of p_i and q_i as below.

$$IR(p,q) = \sum_{i=1}^{T} p_i \log \frac{2 * p_i}{p_i + q_i} + \sum_{i=1}^{T} q_i \log \frac{2 * q_i}{p_i + q_i} \tag{4.3}$$

The IR can be transformed into a similarity measure using the following equation (Dagan, Lee, & Pereira, 1997):

$$SIM(p,q) = 10^{-\delta\ IR(p,q)} \tag{4.4}$$

The Hellinger distance between two distributions is another option that allows avoiding the shortcomings of the KL distance.

$$HD(p,q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{T} (\sqrt{p_i} - \sqrt{q_i})^2} \tag{4.5}$$

The Hellinger distance varies from 0 to 1 and is defined for all values of p_i and q_i. A value of 1 means the distance is maximum and thus the distributions are very different. A value of 0 means the distributions are very similar. We can transform the Hellinger distance into a similarity measure by subtracting it from 1 such that a zero distance means a large similarity score and vice versa.

Lastly, we used the Manhattan distance between distributions p and q as defined below.

$$MD(p,q) = \sum_{i=1}^{T} |p_i - q_i| \tag{4.6}$$

MD is symmetric, defined for any values of p and q, and ranges between 0 and 2. We can divide MD by 2 and subtract from 1 to transform it into a normalized similarity measure.

We further refined the above proposals for computing the similarity of two documents. Besides using the similarity of distributions over topics we can also account for distributions for similarity of topics. To compute the distance between two topics using their distributions over words (ϕ_{t1} and ϕ_{t2}) we can apply the same methods discussed above.

All the results reported in this paper for LDA similarity measures between two documents c and d are computed by multiplying the similarities between the corresponding distribution over topics (θ_d and θ_c) and distribution over words (ϕ_{t1} and ϕ_{t2}).

4.3.3 Experiments and Results

We present results with the proposed LDA-based measures and also, for comparison purposes, results obtained with LSA and the WordNet measures. The results were obtained using the ULPC and MSRP data sets. We followed a training-testing methodology according to which we first trained to learn some parameters of the proposed models after which we used the models on testing data. In our case, we learned a threshold for the text-to-text similarity score above which a pair of sentences is deemed a paraphrase and any score below the threshold means the sentences are not paraphrases. We report performance of the various methods using accuracy (percentage of correct predictions), F-measure (harmonic mean of

precision and recall), and kappa statistics (a measure of agreement between our method outputs and experts' labels while accounting for chance agreement).

Data Sets

The MSRP corpus consists of 5,801 sentence pairs collected from newswire articles, 3,900 of which were labeled as paraphrases by human annotators. The whole set is divided into a training subset (4,076 sentences of which 2,753, or 67.5%, are true paraphrases), and a test subset (1,725 pairs of which 1,147, or 66.5%, are true paraphrases). A simple baseline for the MSRP corpus, the majority baseline when all instances are classified as positive, gives an accuracy and precision of 66.5% and perfect recall. The average number of words per sentence is 17.

The ULPC corpus contains pairs of target-sentence and student response texts. These pairs have been evaluated by expert human raters along 10 dimensions of paraphrase characteristics. We used the Semantic Completeness dimension that measures the semantic equivalence between the target-sentence and the student response on a binary scale, similar to the scale used in the MSRP corpus. From a total of 1,998 pairs, 1,436 (71%) were classified by experts as being paraphrases. The data set is divided into three subsets: training (1,012 instances, 708-304 split of TRUE-FALSE paraphrases), validation (649 instances, 454-195 split), and testing (337 instances, 228-109 split). In the ULPC corpus, the average number of words per sentence is 15.

Generating LSA and LDA Models

An important step in the process of obtaining the LSA vectorial representation is the derivation of the semantic space, i.e., discovering the latent dimensions or concepts, from a large enough corpus. In our work, we experimented with an LSA space of 300 dimensions computed from the TASA corpus (compiled by Touchstone Applied Science Associates), a balanced collection of representative texts from various genres (science, language arts, health, economics, social studies,

business, and others). The TASA corpus contains 10,937,986 words with a vocabulary size of 91,897 after removing stop words.

LDA models are also generated from the TASA corpus. We removed stop words and used default values for the LDA hyper-parameters as described in Section 4.3.1. The models are generated using JGibbLDA[1], a Java implementation of LDA using Gibbs sampling.

Results

We generated a number of LDA models from the TASA corpus. As mentioned earlier, a LDA model assumes the existence of latent topics underlying texts where each topic is a distribution over words. We illustrate three sample topics from one of our LDA models in 4.1. Each topic has a list of words ranked by a probability score. Alternatively, each word has a certain contribution towards each topic and the table presents the top contributing words towards the corresponding topic. In Topic 2, words related to business appear at the top of the list whereas words related to politics appear at the top of Topic 8. Sometimes topics are hard to interpret by humans. Topic 7 is an example of such a topic where, unlike Topic 2 and Topic 8, it is hard to interpret what semantic information the topic captures.

After LDA models were generated, we applied them to infer the topic proportions in the sentences from the paraphrase corpus. To better illustrate this approach, we use the example below, which is instance #23 in the MSRP test data.

Text A: Senator Clinton should be ashamed of herself for playing politics with the important issue of homeland security funding, he said.

Text B: She should be ashamed of herself for playing politics with this important issue, said state budget division spokesman Andrew Rush.

Table 4.2 shows topic assignment for each of the non-stop words in the two sentences from instance #23 in the MSRP test data when using one of our LDA

[1]http://jgibblda.sourceforge.net/

Table 4.1: Examples of topics and distributions overs words in three topics (top 10 words are shown for each topic).

Topic 2		**Topic 7**		**Topic 8**	
Word	**Prob**	**Word**	**Prob**	**Word**	**Prob**
number	0.014	day	0.029	states	0.020
money	0.012	good	0.021	world	0.019
system	0.010	thought	0.0194	united	0.015
business	0.009	school	0.017	government	0.013
information	0.009	home	0.017	american	0.012
special	0.009	children	0.015	state	0.012
set	0.009	father	0.014	war	0.011
job	0.009	knew	0.013	power	0.009
amount	0.009	told	0.0131	president	0.008
general	0.008	hard	0.011	groups	0.007

models. Each word in the sentence is sampled from a topic. For instance, the words *senator*, *issue*, *homeland*, and *funding* are sampled from topic 8.

Table 4.2: Topic assignment for instance #23 in MSRP test data.

Word	**Topic**	**Word**	**Topic**
senator	8	ashamed	1
Clinton	3	playing	7
ashamed	1	politics	8
playing	7	important	10
politics	8	issue	8
important	9	state	8
issue	8	budget	2
homeland	8	division	11
security	2	spokesman	1
funding	8	Andrew	3
		Rush	5

Word-to-Word Similarities

Having LSA and LDA models ready, we computed and compared the semantic similarities between 41,037 word pairs using WordNet, LSA, and LDA. The word pairs are generated from the MSRP data set. Equation 4.1 is used to compute the LDA based word-to-word similarity. To have a reasonable comparison, we used the LDA-model with 300 topics and the LSA model with 300 semantic

dimensions while computing the word similarities. For WordNet, we used two semantic similarity measures: LIN for nouns and verbs (Lin, 1998), and LESK for adjectives and adverbs (Banerjee & Pedersen, 2003).

We present results for 10 word pairs in Table 4.3. Based on the analysis of results, we would like to make a number of interesting observations. First of all, LDA- and LSA-based measures were able to compute similarities between two words with different parts-of-speech. Moreover, LDA and LSA measures were able to compute the semantic similarities between words even if the words were not in the dictionary. The reason behind these benefits is because LDA and LSA measures exploit the statistical properties of the words rather than a predefined hierarchy.

Moreover, it was found that a single method was not superior enough to compute the similarity scores for all word pairs. Alternatively, WordNet based methods were good for some words pairs while the LDA based method was good for other pairs and so was the case for the LSA based method. For example, the similarity between *panel-board* word pair was appropriately computed by WordNet and LDA measures but not by the LSA measure. For the *kidney-dialysis* pair, LSA performed better than WordNet and LDA. Thus, a very interesting future work

Table 4.3: Word-to-word similarity scores for ten pairs of words using WordNet, LSA and LDA

w_1	w_2	**WordNet**	**LSA**	**LDA**
panel	board	1.0000	0.1534	0.9277
man	inventor	0.2283	0.0008	0.8136
revenue	attorney	0.0000	0.0899	0.0002
financial	due	0.0000	0.0000	0.3635
kidney	dialysis	0.0000	0.4282	0.0070
say	move	0.7072	0.0000	0.0000
percent	figure	0.3712	0.0767	1.0000
dog	animal	0.7597	0.0491	0.3725
trade	market	0.6828	0.0619	0.6966
refund	payment	0.8588	0.2242	0.9036

would be to combine all the three methods in some ways to compute the word-to-word similarity scores.

Paraphrase Detection

The objective of paraphrase detection is to test whether two short sentences are semantically equivalent in meaning or not. In this experiment, we applied the previously proposed methods to compute similarity scores between short sentences and use the scores to predict whether they are paraphrases or not.

Table 4.4: Results on the MSRP test data

Method	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**Kappa**
Baseline	66.55	66.53	100	79.90	0.22
LSA	73.56	75.34	89.53	81.83	34.61
LSA Greedy	72.86	75.50	87.61	81.11	33.89
LSA Optimal	73.04	76.72	85.35	80.80	35.95
LDA-IR	66.49	66.55	99.73	79.83	0.34
LDA-Hellinger	65.73	66.64	97.03	79.02	0.86
LDA-Manhattan	66.66	66.60	100	79.95	0.68
LDA-Greedy	71.71	76.21	83.52	79.70	33.36
LDA-Optimal	72.98	76.74	85.17	80.74	35.90
WordNet-Greedy	73.56	76.23	87.53	81.49	36.00
WordNet-Optimal	73.56	75.76	88.57	81.67	35.28

We started by using an LSA model with 300 dimensions and an LDA model with 300 topics. As before, for WordNet, we chose LIN (Lin, 1998) for nouns and verbs, and LESK (Banerjee & Pedersen, 2003) for adjectives and adverbs.

A summary of result for MSRP data set is presented in the Table 4.4. These are results on MSRP test data obtained using a threshold for similarity that corresponds to the threshold learned from training data that led to the best accuracy. The threshold varied from method to method. The results obtained using the word-to-word similarity measures are labeled Greedy and Optimal in Table 4.4. The row labeled LSA shows results obtained when text-level LSA vectors were used, as explained earlier. Similarly, rows with labels WordNet-Greedy and WordNet-Optimal represent the results corresponding to the WordNet-based

Table 4.5: Results on the ULPC test data

Method	**Accuracy**	**Precision**	**Recall**	**F-Measure**	**Kappa**
Baseline	67.65	67.65	100	80.70	0
LSA	77.74	77.03	95.6	85.32	41.43
LSA Greedy	76.85	75.68	96.92	85	37.54
LSA Optimal	75.07	77.90	88.15	82.71	38.63
LDA-IR	67.65	67.65	100	80.70	0
LDA-Hellinger	67.65	67.65	100	80.70	0
LDA-Manhattan	67.65	67.65	100	80.70	0
LDA-Greedy	76.85	75.86	96.49	84.94	37.89
LDA-Optimal	75.96	74.09	99.12	84.80	32.66
WordNet-Greedy	76.55	77.49	92.10	84.16	40.28
WordNet-Optimal	77.44	77.53	93.85	84.92	41.78

measures. The Baseline method indicates performance when labeling all instances with the dominant label of a true paraphrase. The rest of the rows in the table show results when the text-to-text similarity measures based on various distribution distances were used: IR (Information Radius), Hellinger, and Manhattan.

The LDA-Optimal yielded competitive results on MSRP data set. It provided the best precision score. As noted from Table 4.4, the text-to-text similarity measures based on distribution distances performed close to chance. The problem seemed to be rooted in the relative size of texts compared to the number of the topics in the LDA model. As mentioned, we used 300 topics LDA model in this setting. The average sentence size in MSRP (after removing stopwords) is 10.3 for training data and 10.4 for testing data. That means that in a typical sentence most of the 300 topics would not be assigned to any word leading to very similar topic distributions over the entire set of 300 topics. Even if the probability for topics that were not assigned to a word in a sentence was set to 0, the distance between two values of 0 was 0 which meant the distributions were quite similar.

Next, we compared the performances of the methods in the ULPC corpus. The summary of the results are shown in Table 4.5. Here also, among the

LDA-based measures, LDA-Optimal and LDA-Greedy yielded competitive results with that of LSA- and WordNet-based measures. LDA's distribution distance based measures perform poorly on the ULPC corpus as well. As described above, the problem is due to the sparseness of topics assigned to the words in the short sentences.

4.4 Short Text Similarity using Regression

The importance of semantic similarity in NLP is highlighted by the diversity of data sets and shared task evaluation campaigns over the last decade (Agirre, Cer, Diab, Gonzalez-Agirre, & Guo, 2013; Agirre et al., 2014; Agirre, Diab, Cer, & Gonzalez-Agirre, 2012; Dolan et al., 2004; Rus, Banjade, & Lintean, 2014) and by many uses such as in text summarization (Aliguliyev, 2009) and student answer assessment (Niraula, Banjade, Ştefănescu, & Rus, 2013; Rus & Lintean, 2012).

The plethora of measures available in the literature for measuring short texts semantic similarity suggests that no single method is capable of adequately quantifying the semantic similarity between the texts. Therefore, we hypothesize that combining diverse approaches provide a better result. With this hypothesis in mind, we propose a regression-based method to predict a semantic similarity score between two short texts.

Features for the regression include different sentence-to-sentence similarity scores, presence of negation cues, lexical overlap measures etc. The sentence-to-sentence similarity scores were calculated using word-to-word similarity methods and optimal word and chunk alignments. We describe these methods below.

4.4.1 Word-to-Word Similarity

We used knowledge based, corpus based, and hybrid methods to compute word-to-word similarity. From the knowledge based category, we used WordNet (Miller, 1995) based similarity methods from SEMILAR Toolkit (Rus, Lintean,

Banjade, Niraula, & Stefanescu, 2013) which include Lin (Lin, 1998), Lesk (Banerjee & Pedersen, 2003), Hso (Hirst & St-Onge, 1998), Jcn (Jiang & Conrath, 1997), Res (Resnik, 1995), Path, Lch (Leacock & Chodorow, 1998), and Wup (Wu & Palmer, 1994).

In corpus based category, we used LSA models[2] generated from the whole Wikipedia articles as described in Stefanescu et al. (2014a). We also used pre-trained Mikolov word representations (Mikolov et al., 2013)[3] and GloVe word vectors (Pennington, Socher, & Manning, 2014)[4]. In these cases, each word was represented as a vector encoding and the similarity between words were computed as cosine similarity between corresponding vectors. We exploited the lexical relations between words, i.e., synonymy and antonymy, from WordNet 3.0. As such we computed similarity scores between two words a and b as:

$$sim(a,b) = \begin{cases} 1, \text{ if a and b are synonyms} \\ 0, \text{ if a and b are antonyms} \\ \dfrac{\mathbf{A}.\mathbf{B}}{|\mathbf{A}||\mathbf{B}|}, \text{ otherwise} \end{cases}$$

where $\mathbf{A}$ and $\mathbf{B}$ are vector representations of words a and b respectively.

In hybrid approach, we developed a new word-to-word similarity measure (hereafter referred as Combined-Word-Measure) by combining the WordNet-based similarity methods with corpus based methods (using Mikolov's word embeddings and GloVe vectors) by applying Support Vector Regression(SVR). We did experiments with the recently published word similarity dataset called Simlex-999 (Hill et al., 2014) and achieved correlation (r) of 0.601 with human judgment (Banjade, Maharjan, Niraula, Rus, & Gautam, 2015).

[2]Models available at http://semanticsimilarity.org
[3]Downloaded from http://code.google.com/p/word2vec/
[4]Downloaded from http://nlp.stanford.edu/projects/glove/

4.4.2 Sentence-to-Sentence Similarity

We applied three different approaches to compute sentence-to-sentence similarity.

Optimal Word Alignment Method

We first computed the similarity of word pairs (all possible combinations) using all similarity methods described in Section 4.4.1. The similarity score less than 0.3 (empirically set threshold), was reset to 0 in order to avoid noisy alignments. Then the words were aligned as discussed in Section 4.1.3 so that the overall alignment score between the full sentences was maximum. Once the words were aligned optimally, we calculated the sentence similarity score as the sum of the word alignment scores normalized by the average length of the sentence pair.

Optimal Chunk Alignment Method

We created chunks and aligned them to calculate sentence similarity as in Stefanescu, Banjade, and Rus (2014b) and applied optimal alignment twice. First, we applied optimal alignment of words in two chunks to measure the similarity of the chunks. As before, word similarity threshold was set to 0.3. We then normalized chunk similarity by the number of tokens in the shorter chunk such that it assigned higher scores to pairs of chunks such as *physician* and *general physician*. Second, we applied optimal alignment at chunk level in order to calculate the sentence level similarity. We used chunk-to-chunk similarity threshold 0.4 to prevent noisy alignments. In this case, however, the similarity score was normalized by the average number of chunks in the given texts pair. All threshold values were set empirically based on the performance on the training set.

Resultant Vector Based Method

In this approach, we combined vector based word representations to obtain sentence level representations through vector algebra. We added the vectors corresponding to content words in each sentence to create a resultant vector for each

sentence and the cosine similarity was calculated between the resultant vectors. We used word vector representations from Wiki LSA, Mikolov and GloVe models.

For a missing word, we used vector representation of one of its synonyms obtained from the WordNet. To compute the synonym list, we considered all senses of the missing word given its POS category.

4.4.3 Features for Regression

1. Similarity scores using optimal alignment of words where word-to-word similarity was calculated using vector based methods using word representations from Mikolov, GloVe, LSA Wiki models and Combined-Word-Measure which combines knowledge based methods and corpus based methods.

2. Similarity score using optimal alignment of chunks where word-to-word similarity scores were calculated using Mikolov's word representations.

3. Similarity scores based on the resultant vector method using word representations from Mikolov, GloVe, and LSA Wiki models.

4. Noun-Noun, Adjective-Adjective, Adverb-Adverb, and Verb-Verb similarity scores and similarity score for other words using optimal word alignment and Mikolov's word representations.

5. Multiplication of noun-noun similarity score and verb-verb similarity score (scores calculated as described in 4).

6. Whether there was any antonym pair present.

7. $\frac{|C_{i1} - C_{i2}|}{C_{i1} + C_{i2}}$ where C_{i1} and C_{i2} are the counts of $i \in \{$all tokens, adjectives, adverbs, nouns, and verbs$\}$ for sentence 1 and 2 respectively.

8. Presence of adjectives and adverbs in first sentence, and in the second sentence.

Table 4.6: Summary of training data

Data set	Count	Release time
SMTnews	351	STS2012-Test
Headlines	1500	STS2013-Test
Deft-forum	423	STS2014-Test
Deft-news	299	STS2014-Test
Images	749	STS2014-Test

9. Unigram overlap with synonym check, bigram overlap and BLEU score (Papineni, Roukos, Ward, & Zhu, 2002).

10. Presence of negation cue (e.g., no, not, never) in either of sentences.

11. Whether one sentence was a question while the other was not.

12. Total number of words in each sentence. Similarly, the number of adjectives, nouns, verbs, adverbs, and others, in each sentence.

4.4.4 Experiments and Results

We trained and evaluated the proposed regression-based method by using data provided by SemEval shared task on semantic textual similarity focused on English STS(Agirre et al., 2015). The English STS subtask was about assigning a similarity score between 0 and 5 to pairs of sentences; a score of 0 meaning the sentences are unrelated and 5 indicating they are equivalent.

Data: For training, we used data released in previous shared tasks (summarized in Table 4.6). We selected data sets that include texts from different genres. However, some others were not included in the training set. For instance, Tweet-news were not included as they were quite different from most of other texts and special treatment might be needed. Being more biased towards overlapping text, such as MSRPar (Rus, Banjade, & Lintean, 2014), was also a concern.

The test set included data (sentence pairs) from Answers-forums (375), Answers-students (750), Belief (375), Headlines (750), and Images (750).

Preprocessing: We removed stop words, labeled each word with Part-of-Speech (POS) tag and lemmatized them using Stanford CoreNLP Toolkit (Manning et al., 2014). Some text pairs (most notably in student answers and forum data) had many commonly used words (many of them would be treated as stop words). So, we revised the stopword list by removing some words, we thought more informative, such as same form the list. We did spelling corrections in student answers and forum data as the possibility of spelling errors is comparatively high in these texts. We used Jazzy tool (Idzelis, 2005) with WordNet dictionary for spelling correction. Moreover, in student answers data, we found that the symbol A (such as in bulb A and node A) typed in lowercase was incorrectly labeled as a determiner 'a' by the POS tagger. So, we applied a rule to correct it. If the token after 'a' is not an adjective, adverb, or noun, or the token is the last token in the sentence, we changed its type to noun (NN). Additionally, we removed comma from the numbers. We then created chunks as described by Stefanescu et al. (2014b).

Regression: We generated various features as described in Section 4.4.3 and applied regression methods in three different settings. In the first run (R1), all features were used in SVM Regression with Radial Basis Function kernel. The second run (R2) was same as R1 except that the features in R2 did not include the count features (i.e., features in 12). In the third run (R3), we used features same as R2 but applied linear regression instead.

For SVR, we used LibSVM library (Chang & Lin, 2011) in Weka (Hall et al., 2009) and for the linear regression we used Weka's implementation. The 10-fold cross validation results (r) of three different runs with the training data were 0.7734 (R1), 0.7662 (R2), and 0.7654 (R3).

The results on the test set have been presented in Table 4.7. Though R1 had the highest correlation score in a 10-fold cross validation process using the training data, the results of R2 and R3 on the test data were consistently better than the

Table 4.7: Results of our submitted runs on test data.

Data set	**Baseline**	**R1**	**R2**	**R3**
Ans-forums	0.445	0.526	0.694	0.677
Ans-students	0.664	0.725	0.744	0.735
Belief	0.651	0.631	0.751	0.722
Headlines	0.531	0.813	0.807	0.812
Images	0.603	0.858	0.864	0.857
Mean	0.587	0.743	**0.784**	0.776

results of R1. It suggests that absolute count features used in R1 tend to overfit the model. The weighted mean correlation of R2 was 0.784 - the best among our three runs and ranked 10th among 74 runs submitted by 29 participating teams. The correlation score was very close to the results of other best performing systems. However, the correlation scores of answer-forum, answer-students, and belief data were found to be lower than those of headlines and images data. The reason might be the texts in the former data being not well-written as compared to the latter. Also, more contextual information is required to fully understand them.

4.5 Interpretable Semantic Textual Similarity

As we saw, the task of semantic similarity measures the degree of semantic equivalence between two texts in terms of a score. This is very useful for assessing student responses but it is not sufficient since it fails to explain the reasons behind being similar, related or unrelated. Furthermore, it does not tell the type of semantic relations that exist among the constituents such as words or chunks. To have a concrete idea, consider an example below showing a student's response and the corresponding expected answer (square brackets enclose chunks) :

Student Answer: [Newton's laws of motion] [apply].

Expected Answer:[Newton's third law] [is relevant] [in the collision].

Relations :

Specific: [Newton's laws of motion] $\Leftrightarrow$ [Newton's third law]

Equivalent: [applies] ⇔ [is relevant]

No Alignment: []⇔[in the collision]

In this example, chunks in the student response are aligned to the chunks in the expected answer. The type of relations (equivalent, specific and no align) are also provided. By doing this, we get interpretation of the alignments and similarities which is always better to have than a single holistic score provided by semantic textual similarity. For instance, the holistic score does not indicate that the student is giving a vague answer (laws of motion) instead of a specific one (the third law). Therefore, finding reasons and explicit relations among their constituents in a paired texts (also known as *Interpretable Semantic Textual Similarity; iSTS*) would enable a meaningful interpretation of the similarity scores which can be exploited for better follow-up question and feedback generation in ITS. For instance, in the previous example, as student's response is vague, we can ask this follow up question: *Can you tell the specific Newton's law ?*

There are some works in literature in this direction. Brockett (2007) and Rus et al. (2012) produced datasets where corresponding words (or multiword expressions) were aligned and in the later case their semantic relations were explicitly labeled. Below, we present our iSTS system that finds the type of relations among their constituents and the similarity scores based on our submission for SemEval-2015 shared task (Banjade, Niraula, et al., 2015).

4.5.1 A Rule Based System for iSTS

Input to our system is a pair of sentences with their chunking information (gold chunks). The task then is to map chunks of the first sentence to those from the second by assigning different relations and scores based on a set of rules.The alignment is restricted to one-to-one for simplicity. Further details about the task including relation types and the evaluation criteria can be found at Agirre et al. (2015).

Type of Alignments

For each alignment we need to decide its type i.e., a semantic relation and a similarity score (0 − unrelated, 5 − equivalent). The list of semantic relations are listed below.

- EQUI: This relation is assigned when two chunks are semantically equivalent in meaning in the context.
- OPPO: This relation is assigned when two chunks are in opposition to each other in the context.
- SPE1 and SPE2: When a chunk in the first chunk is more specific than the corresponding chunk in the second sentence, a SPE1 relation is assigned. SPE2 is defined similarly.
- SIMI: This relation is assigned when the chunks have similar meanings but no EQUI, OPPO, SPE1 and SPE2 relations between them.
- REL: This relation is assigned when the chunks have related meanings but no EQUI, OPPO, SPE1, SPE2, and SIMI relations.
- ALIC: Because of one-to-one alignment restriction, a chunk may not get a chance to pair with a chunk in the next sentence. In such case, an ALIC relation is assigned to the chunk that couldn't pair with the another chunk in the pair.
- NOALI: When a chunk in a sentence has no corresponding chunks in another sentence, NOALI relation is assigned.

Further details about the task including relation types and the evaluation criteria can be found in Agirre et al. (2015).

Preprocessing

The system performs stop word marking, POS tagging, lemmatization, and named-entity recognition in the preprocessing steps. It also uses lookups for data normalization as well as synonym, antonym and hypernym relations.

For data normalization, we manually constructed a lookup table for commonly used words by mapping them to standard values. For instance, %, *percent*, *percentage* all map to *pc*. For synonym lookup, we created a strict synonym lookup file using WordNet. Similarly, an antonym lookup file was created by building an antonym set for a given word from its direct antonyms and their synsets. We further constructed another lookup file for strict hypernyms.

Rules

In this section, we describe the rules used for chunk alignments and scoring. The scores given by each rule are highlighted.

Conditions: We define below a number of conditions for a given chunk pair that might be checked before applying a rule.

C_1: One chunk has a conjunction and other does not

C_2: A content word in a chunk has an antonym in the other chunk

C_3: A word in either chunk is a NUMERIC entity

C_4: Both chunks have LOCATION entities

C_5: Any of the chunks has a DATE/TIME entity

C_6: Both chunks share at least one content word other than noun

C_7: Any of the chunks has a conjunction

Next, we define a set of rules for each relation type. For aligning a chunk pair (A, B), these rules are applied in order of precedence as NOALIC, EQUI, OPPO, SPE, SIMI, REL, and ALIC. Once a chunk is aligned, it would not be considered for further alignments. Moreover, there is a precedence of rules within each relation type e.g., EQ_2 is applied only if EQ_1 fails and EQ_3 is applied if both

EQ_1 and EQ_2 fail and so on. If a chunk does not get any relation after applying all the rules, a NOALIC relation is assigned. Note that we frequently use *sim-Mikolov*(A, B) to refer to the similarity score between the chunks A and B using Mikolov word vectors as described in Section 4.4.2.

NOALIC Rules

NO_1: If a chunk to be mapped is a single token and is a punctuation, assign NOALIC

EQUI Rules

EQUI Rules $EQ_1 - EQ_3$ are applied unconditionally. The rest rules ($EQ_4 - EQ_5$) are applied only if none of conditions C_1 - C_5 are satisfied.

EQ_1 - Both chunks have same tokens (**5**) - e.g., to compete ⇔ To Compete

EQ_2 - Both chunks have same content words (**5**) - e.g., in Olympics ⇔ At Olympics

EQ_3 - All content words match using synonym lookup (**5**) - e.g., to permit ⇔ Allowed

EQ_4 : All content words of a chunk match and unmatched content word(s) of the other chunk are all of proper noun type (**5**) - e.g., Boeing 787 Dreamliner ⇔ on 787 Dreamliner

EQ_5 : Both chunks have equal number of content words and $sim - Mikolov(A, B) > 0.6$ (**5**) - e.g., in Indonesia boat sinking ⇔ in Indonesia boat capsize

OPPO Rules

OPPO rules are applied only when none of C_3 and C_7 are satisfied.

OP_1: A content word in a chunk has an antonym in the other chunk (**4**) - e.g., in southern Iraq ⇔ in northern Iraq

SPE Rules

SP_1: If chunk A but B has a conjunction and A contains all the content words of B

then A is SPE of B (**4**) - e.g., Angelina Jolie ⇔ Angelina Jolie and the complex truth.

SP_2: If chunk A contains all content words of chunk B plus some extra content words that are not verbs, A is a SPE of B or vice-versa. If chunk B has multiple SPEs, then the chunk with the maximum token overlap with B is selected as the SPE of B. (**4**) - e.g., Blade Runner Pistorius ⇔ Pistorius.

SP_3: If chunks A and B contain only one noun each say n_1 and n_2 and n_1 is hypernym of n_2, B is SPE of A or vice versa (**4**) - e.g., by a shop ⇔ outside a bookstore.

SIMI Rules

SI_1: Only the unmatched content word in each chunk is a CD type(**3**)-e.g., 6.9 magnitude earthquake ⇔ 5.6 magnitude earthquake

SI_2: Each chunk has a token of DATE/TIME type (**3**)- e.g., on Friday ⇔ on Wednesday

SI_3: Each chunk has a token of LOCATION type (**3**) - e.g., Syria ⇔ Iraq

SI_4: When both chunks share at least one noun then assign **3** if $sim\text{-}Mikolov(A, B) >= 0.4$ and **2** otherwise. - e.g., Nato troops ⇔ NATO strike

SI_5: This rule is applied only if C_6 is not satisfied. Scores are assigned as : (i) **4** if $sim\text{-}Mikolov(A, B) \in [0.7, 1.0]$ (ii) **3** if $sim\text{-}Mikolov(A, B) \in [0.65, 0.7)$ (iii) **2** if $sim\text{-}Mikolov(A, B) \in [0.60, 0.65)$

REL Rules

RE_1: If both chunks share at least one content word other than noun then assign REL relation. Scores are assigned as follows : (i) **4** if $sim\text{-}Mikolov(A, B) \in [0.5, 1.0]$ (ii) **3** if $sim\text{-}Mikolov(A, B) \in [0.4, 0.5)$ (iii) **2** otherwise. e.g., to Central African Republic ⇔ in Central African capital

ALIC Rules

AL_1: If a chunk in a sentence X (C_x) is not aligned yet but has a chunk in another

Table 4.8: F_1 scores for *Images* and *Headlines* data sets. A, T and S refer to Alignment, Type, and Score respectively.

	Run	A	T	S	T+S
Headlines	Baseline	0.844	0.555	0.755	0.555
	R_1	**0.898**	0.654	0.826	0.638
	R_2	0.897	0.655	**0.826**	0.640
	R_3	0.897	**0.666**	0.815	**0.642**
Images	Baseline	0.838	0.432	0.721	0.432
	R_1	**0.887**	**0.614**	0.787	0.584
	R_2	0.880	0.585	0.781	0.561
	R_3	0.883	0.603	0.783	0.575

pair-sentence Y (C_y) that is already aligned and has $sim\text{-}Mikolov(C_x, C_y) >= 0.6$, assign ALIC relation to C_x with a score of (**0**).

4.5.2 Experiments and Results

We applied above mentioned rules in the training data set provided by the SemEval-2015 by varying thresholds for *sim-Mikolov* scores and selected the thresholds that produced the best results in the training data set. Since three runs were allowed to submit, we defined them as follows:

Run 1 (R_1) : Applied full set of rules with limited stop words (375 words)

Run 2 (R_2) : Same as R_1 but with extended stop words (686 words).

Run 3 (R_3) : Applied full set of rules with extended stop words but with one exception: EQ_4 was modified such that it would apply when unmatched content words of the bigger chunk were of noun rather than proper noun type.

There were 16 runs submitted by 7 different teams. The results corresponding to our three runs and that of the baseline are presented in Table 4.8. The highlighted scores were the best scores among all the submissions from the competing teams. In Headlines test data, at least one of our runs outperformed the rest competing submissions in all evaluation metrics. In Images test data, R_1 was the best in alignment and type metrics. Our submissions were among the top performing submissions for score and type+score metrics.

R_3 performed better among all runs in case of Headlines data in overall. This was chiefly due to modified EQ_4 rule which reduced the number of incorrect EQUI alignments. We also observed that performance of our system was least affected by size of stopword list for Headlines data as both R_1 and R_2 recorded similar F_1-measures for all evaluation metrics. However, R_1 performed relatively better than R_2 in Images data-particularly in correctly aligning chunk relations. It could be that images are described mostly using common words and thus were filtered by R_2 as stop words.

4.6 Discussions and Conclusions

In this chapter, we addressed the problem of computing semantic similarity between two short texts which are typically the student responses. Specifically, we proposed two types of LDA-based semantic similarity measures. The first measure relied on word-to-word similarity using the dot-product between topic vectors followed by using greedy and optimal matching methods. The second measure computed the divergence between two distributions corresponding to the texts and then converted them to similarity scores. Based on the evaluations on two standard paraphrase detection corpora the MSRP and the ULPC, it was found that word-to-word LDA-based measure was competitive with LSA and WordNet-based measures for detecting paraphrases. However, the divergence-based similarity measures were not effective for computing semantic similarity between short text due to topic sparseness problem.

Next, we proposed a regression-based approach to predict more fine grain (between 0 to 5) semantic textual semantic similarity of given sentence pairs. Our system rivaled with top performing systems against the standard test data set provided by SemEval-2015 shared task. The system was very competitive to the top performing approaches in SemEval-2015 shared task.

Lastly, we presented a system for interpretable semantic textual similarity. It

relied on a set of rules blended with similarity features in order to assign the labels and scores for the chunk-level relations. Our system was among the top performing systems in this subtask in SemEval-2015 shared task. Since we relied on the gold chunks, the immediate future works is to automatically generate such chunks by using sequence tagging techniques such as conditional random fields.

Chapter 5

Conclusions and Future Directions

This chapter summarizes the contributions of this dissertation and presents some potential future directions.

5.1 Conclusions

The popularity of dialog-based intelligent tutoring systems has been rising due to their effectiveness at inducing learning gains in students. However, scaling of such systems is a big problem as they demand significant manual efforts for dialog generation. This dissertation addressed many challenges that hinder the scaling of the systems.

First, we presented novel and efficient approaches to generate and rank cloze and open-cloze questions respectively (Chapter 2). We generated cloze questions by mining student tutor interaction logs. We proposed an active learning approach to rank automatically generated open-cloze questions.

Second, we proposed a machine learning approach to resolve pronouns in student responses (Chapter 3). The approach was very accurate on resolving the pronouns in student answers.

Third, we conducted experiments to quest a better approach for computing semantic textual similarity between two short texts (Chapter 4). We focused on short texts because student responses were typically short in length. The problem of computing semantic textual similarity is very crucial to dialog generation since incorrect assessments lead to incorrect feedback which can lower confidence of a student with a tutor and hamper the effectiveness of learning. A regression based method was found to be efficient for this task.

Lastly, we proposed a system for interpretable semantic textual similarity (Chapter 4) that can explain the reason behind the holistic score provided by

semantic textual similarity methods. The system was one of the best systems in SemEval-2015 shared task.

5.2 Future Directions

This dissertation contributed some novel research works to the literature of automatic question generation and student answer assessment. These are the two major sub-problems of automatic dialog generation problem in the context of intelligent tutoring system. There are still more challenges to be addressed towards this bigger goal. We discuss below some of the possible future directions.

- Deep Question Generation: Generating deep questions is one of the challenging problems to be solved. Here is an example:
  Sentence: *Newton's third is applicable in this context.*
  Relatively easier questions to generate:
  (a) Which law is applicable in this context ?
  (b) Which Netwon's law is applicable in this context ?
  (c) Is Netwon's third law applicable in this context ?
  Harder to generate:
  (d) State a principle that can be applied in the given context.

  The first three questions (a-c) are relatively easier to generate by using systems such as Heilman and Smith (2009) and Mazidi and Nielsen (2014b). The primary reason for calling simple is because they rely on sentence transformation, parsing, Named-entity recognition and semantic role labeling, which are well-studied problems in NLP. The fourth question (d) is harder to generate compared to the rest because it needs deep semantic understanding of the sentence. For instance, in order to generate the fourth question , we must know that Newton's third law is a type of "principle". To do this, we must encode knowledge in some form of semantic graph and do semantic parsing of sentence with this knowledge. Although some existing semantic

networks such as Freebase(http://www.freebase.com) and DBPedia (http://dbpedia.org) contain some general knowledge, their domain coverage and depth is pretty low for question generation. Therefore, we assumed in this dissertations that deep questions will be provided by experts. Generating such questions automatically would further help scaling of the system, a potential direction for future.

- Feedback and Follow-up Question Generation: We addressed the interpretation of textual similarity in Section 4.5. As discussed in that section, this work can further be exploited to generate feedback and follow-up question generation.

- Dialog Management: Dialog management is a core part of a dialog system. A dialog manager decides what a dialog system should do for a given user response to maximize the user's goal. In the context of ITS, given a student answer, what feedback to give and what question to ask him next are few roles of the dialog management. As such, our proposed solutions for automatically assessing student responses and automatically generating questions will play vital roles for dialog managers. A future work would therefore is to exploit these contributions for dialog management.

- Dialog Act Classification, Natural Language Generation and Grounding: These are the other important areas of dialog management. Although, these are well-studied for spoken dialog systems, they are still in premature state for intelligent tutoring systems due to the differences between the two systems.

References

Agarwal, M., & Mannem, P. (2011). Automatic Gap-fill Question Generation from Text Books. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 56–64).

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., ... Wiebe, J. (2015, June). SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015).* Denver, CO: Association for Computational Linguistics.

Agirre, E., Baneab, C., Cardiec, C., Cerd, D., Diabe, M., Gonzalez-Agirrea, A., ... Wiebeg, J. (2014). SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. *SemEval 2014*, 81.

Agirre, E., Cer, D., Diab, M., Gonzalez-Agirre, A., & Guo, W. (2013). sem 2013 Shared Task: Semantic Textual Similarity, Including a Pilot on Typed-similarity. In *In* SEM 2013: The Second Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics.*

Agirre, E., Diab, M., Cer, D., & Gonzalez-Agirre, A. (2012). Semeval-2012 Task 6: A Pilot on Semantic Textual Similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics* (pp. 385–393).

Ali, H., Chali, Y., & Hasan, S. A. (2010). Automation of Question Generation from Sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation.*

Aliguliyev, R. M. (2009). A New Sentence Similarity Measure and Sentence Based Extractive Technique for Automatic Text Summarization. *Expert Systems with Applications*, *36*(4), pp. 7764–7772.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive Tutors: Lessons Learned. *The Journal of the Learning Sciences*, *4*(2), pp.

167–207.

Arregi, O., Ceberio, K., Daz de Illarraza, A., Goenaga, I., Sierra, B., & Zelaia, A. (2010). A First Machine Learning Approach to Pronominal Anaphora Resolution in Basque. In A. Kuri-Morales & G. Simari (Eds.), *Advances in Artificial Intelligence - IBERAMIA 2010* (Vol. 6433, pp. 234-243). Springer Berlin Heidelberg.

Banerjee, S., & Pedersen, T. (2003). Extended Gloss Overlaps As a Measure of Semantic Relatedness. In *IJCAI* (Vol. 3, pp. 805–810).

Banjade, R., Maharjan, N., Niraula, N. B., Rus, V., & Gautam, D. (2015). Lemon and Tea Are Not Similar: Measuring Word-to-word Similarity Combining Different Methods. In *Proceedings of the 16th International Conference on Intelligent Text Processing and Computational Linguistics.*

Banjade, R., Niraula, N. B., Maharjan, N., Rus, V., Stefanescu, D., Lintean, M., & Gautam, D. (2015). NeRoSim: A System for Measuring and Interpreting Semantic Textual Similarity. In *Proceedings of the Workshop on SemEval, NAACL* (pp. 164–171).

Becker, L., Basu, S., & Vanderwende, L. (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* (pp. 742–751).

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *the Journal of machine Learning research*, *3*, pp. 993–1022.

Bollegala, D., Matsuo, Y., & Ishizuka, M. (2007). Measuring Semantic Similarity Between Words Using Web Search Engines. *WWW*, *7*, pp. 757–766.

Brockett, C. (2007). Aligning the RTE 2006 Corpus. *Microsoft Research*.

Brown, J. C., Frishkoff, G. A., & Eskenazi, M. (2005). Automatic Question Generation for Vocabulary Assessment. In *Proceedings of the Conference on*

Human Language Technology and Empirical Methods in Natural Language Processing (pp. 819–826).

Celikyilmaz, A., Hakkani-Tur, D., & Tur, G. (2010). LDA Based Similarity Modeling for Question Answering. In *Proceedings of the NAACL HLT 2010 Workshop on Semantic Search* (pp. 1–9).

Chai, J., Lin, J., Zadrozny, W., Ye, Y., Stys, Budzikowska, M., ... Wolf, C. (2001). The Role of a Natural Language Conversational Interface in Online Sales: A Case Study. *International Journal of Speech Technology*, *4*(3-4), pp. 285–295.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: a Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3), 27.

Chen, J., Schein, A., Ungar, L., & Palmer, M. (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* (pp. 120–127).

Chen, W., Aist, G., & Mostow, J. (2009). Generating Questions Automatically from Informational Text.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (almost) from Scratch. *The Journal of Machine Learning Research*, *12*, pp. 2493–2537.

Corley, C., & Mihalcea, R. (2005). Measuring the Semantic Similarity of Texts. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment* (pp. 13–18).

Curto, S., Mendes, A. C., & Coheur, L. (2011). Exploring Linguistically-rich Patterns for Question Generation. In *Proceedings of the UCNLG+ Eval: Language Generation and Evaluation Workshop* (pp. 33–38).

Dagan, I., Glickman, O., & Magnini, B. (2006). The Pascal Recognising Textual Entailment Challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment* (pp. 177–190). Springer.

Dagan, I., Lee, L., & Pereira, F. (1997). Similarity-based Methods for Word Sense Disambiguation. In *Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistics* (pp. 56–63).

Dolan, B., Quirk, C., & Brockett, C. (2004). Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of the 20th International Conference on Computational Linguistics* (p. 350).

Eason, S. H., Goldberg, L. F., Young, K. M., Geist, M. C., & Cutting, L. E. (2012). Reader–text Interactions: How Differential Text and Question Types Influence Cognitive Skills Needed for Reading Comprehension. *Journal of educational psychology*, *104*(3), 515.

Ferguson, G., Allen, J., Galescu, L., Quinn, J., & Swift, M. (2009). CARDIAC: An Intelligent Conversational Assistant for Chronic Heart Failure Patient Health Monitoring. In *AAAI Fall Symposium Series: Virtual Health Care Interaction (VHI 09), Arlington, VA.*

Gabrilovich, E., & Markovitch, S. (2007). Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *IJCAI* (Vol. 7, pp. 1606–1611).

Gasperin, C., & Briscoe, T. (2008). Statistical Anaphora Resolution in Biomedical Texts. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1* (pp. 257–264).

Graesser, A. C., Jackson, G. T., Mathews, E. C., Mitchell, H. H., Olney, A., Ventura, M., ... others (2003). Why/AutoTutor: A Test of Learning Gains

from a Physics Tutor with Natural Language Dialog. , 1–6.

Graesser, A. C., Lu, S., Jackson, G. T., Mitchell, H. H., Ventura, M., Olney, A., & Louwerse, M. M. (2004). AutoTutor: A Tutor with Dialogue in Natural Language. *Behavior Research Methods, Instruments, & Computers*, *36*(2), pp. 180–192.

Graesser, A. C., Olney, A., Haynes, B. C., & Chipman, P. (2005). AutoTutor: A Cognitive System That Simulates a Tutor Through Mixed-Initiative Dialogue. *Cognitive systems: Human cognitive models in systems design*, 177.

Graesser, A. C., Rus, V., & Cai, Z. (2008). Question Classification Schemes. In *Proc. of the Workshop on Question Generation.*

Graesser, A. C., VanLehn, K., Rosé, C. P., Jordan, P. W., & Harter, D. (2001). Intelligent Tutoring Systems with Conversational Dialogue. *AI magazine*, *22*(4), 39.

Griffiths, T. L., & Steyvers, M. (2004). Finding Scientific Topics. *Proceedings of the National Academy of Sciences of the United States of America*, *101*(Suppl 1), pp. 5228–5235.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA Data Mining Software: an Update. *ACM SIGKDD explorations newsletter*, *11*(1), pp. 10–18.

Heilman, M., & Smith, N. A. (2009). *Question Generation Via Overgenerating Transformations and Ranking* (Tech. Rep.). DTIC Document.

Heilman, M., & Smith, N. A. (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* (pp. 609–617).

Hirst, G., & St-Onge, D. (1998). Lexical Chains As Representations of Context for the Detection and Correction of Malapropisms. *WordNet: An electronic lexical*

database, *305*, pp. 305–332.

Hofmann, T. (1999). Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 50–57).

Hoshino, A., & Nakagawa, H. (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* (pp. 17–20).

Ibrahim, A., Katz, B., & Lin, J. (2003). Extracting Structural Paraphrases from Aligned Monolingual Corpora. In *Proceedings of the Second International Workshop on Paraphrasing-Volume 16* (pp. 57–64).

Idzelis, M. (2005). *Jazzy: The Java Open Source Spell Checker.*

Iordanskaja, L., Kittredge, R., & Polguere, A. (1991). Lexical Selection and Paraphrase in a Meaning-text Generation Model. In *Natural Language Generation in Artificial Intelligence and Computational Linguistics* (pp. 293–312). Springer.

Jiang, J. J., & Conrath, D. W. (1997). Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. *arXiv preprint cmp-lg/9709008*.

Kalady, S., Elikkottil, A., & Das, R. (2010). Natural Language Question Generation Using Syntax and Keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation* (pp. 1–10).

Kuhn, H. W. (1955). The Hungarian Method for the Assignment Problem. *Naval research logistics quarterly*, *2*(1-2), pp. 83–97.

Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (2013). *Handbook of Latent Semantic Analysis*. Psychology Press.

Leacock, C., & Chodorow, M. (1998). Combining Local Context and WordNet Similarity for Word Sense Identification. *WordNet: An electronic lexical*

database, *49*(2), pp. 265–283.

Lin, D. (1998). An Information-theoretic Definition of Similarity. In *ICML* (Vol. 98, pp. 296–304).

Lintean, M. C., Moldovan, C., Rus, V., & McNamara, D. S. (2010). The Role of Local and Global Weighting in Assessing the Semantic Similarity of Texts Using Latent Semantic Analysis. In *FLAIRS Conference.*

Liu, M., Calvo, R. A., & Rus, V. (2012). G-Asks: An Intelligent Automatic Question Generation System for Academic Writing Support. *Dialogue & Discourse*, *3*(2), pp. 101–124.

Mannem, P., Prasad, R., & Joshi, A. (2010). Question Generation from Paragraphs At UPenn: QGSTEC System Description. In *Proceedings of QG2010: The Third Workshop on Question Generation* (pp. 84–91).

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 55–60).

Mazidi, K., & Nielsen, R. D. (2014a). Linguistic Considerations in Automatic Question Generation. In *Proceedings of Association for Computational Linguistics* (pp. 321–326).

Mazidi, K., & Nielsen, R. D. (2014b). Pedagogical Evaluation of Automatically Generated Questions. In *Intelligent Tutoring Systems* (pp. 294–299).

McCarthy, P. M., & McNamara, D. S. (2008). The User-language Paraphrase Challenge. *Retrieved January*, *10*.

Mihalcea, R. (2004). Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions* ()p. 20.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed

Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems* (pp. 3111–3119).

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM*, *38*(11), 39–41.

Mitkov, R. (1999). *Anaphora resolution: The state of the art* (Tech. Rep.). University of Wolverhampton.

Mitkov, R., Evans, R., & Orasan, C. (2002). A New, Fully Automatic Version of Mitkov?s Knowledge-poor Pronoun Resolution Method. In *Computational Linguistics and Intelligent Text Processing* (pp. 168–186). Springer.

Mitkov, R., Ha, L. A., & Karamanis, N. (2006). A Computer-aided Environment for Generating Multiple-choice Test Items. *Natural Language Engineering*, *12*(2), 177–194.

Mitkov, R., Ha, L. A., Varga, A., & Rello, L. (2009). Semantic Similarity of Distractors in Multiple-Choice Tests: Extrinsic Evaluation. In *Proceedings of the Workshop on Geometrical Models of Natural Language Semantics* (pp. 49–56).

Mostow, J., & Chen, W. (2009). Generating Instruction Automatically for the Reading Strategy of Self-Questioning. In *AIED* (pp. 465–472).

Niraula, N. B., Banjade, R., Ştefănescu, D., & Rus, V. (2013). Experiments with Semantic Similarity Measures Based on LDA and LSA. In *Statistical Language and Speech Processing* (pp. 188–199). Springer.

Niraula, N. B., & Rus, V. (2014). A Machine Learning Approach to Pronominal Anaphora Resolution in Dialogue Based Intelligent Tutoring Systems. In *Computational Linguistics and Intelligent Text Processing* (pp. 307–318). Springer.

Niraula, N. B., & Rus, V. (2015). Judging the Quality of Automatically Generated Gap-fill Question Using Active Learning. In *Proceedings of the 10th Workshop*

on Innovative Use of NLP for Building Educational Applications, NAACL (pp. 39–41).

Niraula, N. B., Rus, V., Banjade, R., Stefanescu, D., Baggett, W., & Morgan, B. (2014). The DARE Corpus: A Resource for Anaphora Resolution in Dialogue Based Intelligent Tutoring Systems. In *Proceedings of language resources and evaluation (lrec).*

Niraula, N. B., Rus, V., & Stefanescu, D. (2013). DARE: Deep Anaphora Resolution in Dialogue Based Intelligent Tutoring Systems. In *Proceedings of the 6th international conference on educational data mining (edm 2013)* (pp. 266–267).

Niraula, N. B., Rus, V., Stefanescu, D., & Graesser, A. C. (2014). Mining Gap-fill Questions from Tutorial Dialogues. In *Proceedings of the 7th International Conference on Educational Data Mining* (pp. 265–268).

Ong, J., & Ramachandran, S. (2003). Intelligent Tutoring Systems: Using AI to Improve Training Performance and ROI. *Stottler Henke Associates, Inc. online http://www.shai.com/papers/ITS_using_AI_to_improve_training_performance_and_ROI.pdf*.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 311–318).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, *12*.

Phan, X.-H., Nguyen, L.-M., & Horiguchi, S. (2008). Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections. In *Proceedings of the 17th International Conference on World Wide Web* (pp. 91–100).

Pino, J., Heilman, M., & Eskenazi, M. (2008). A Selection Strategy to Improve Cloze Question Quality. In *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. 9th International Conference on Intelligent Tutoring Systems, Montreal, Canada* (pp. 22–32).

Poesio, M., & Kabadjov, M. A. (2004). A General-purpose, Off-the-shelf Anaphora Resolution Module: Implementation and Preliminary Evaluation. In *Proceedings of LREC.*

Poesio, M., Patel, A., & Di Eugenio, B. (2006). Discourse Structure and Anaphora in Tutorial Dialogues: An Empirical Analysis of Two Theories of the Global Focus. *Research on Language and Computation*, *4*(2-3), pp. 229–257.

Pollock, M., Whittington, C., & Doughty, G. (2000). Evaluating the Costs and Benefits of Changing to CAA. In *Proceedings of the 4th CAA Conference.*

Qiu, L., Kan, M.-Y., & Chua, T.-S. (2004). A Public Reference Implementation of the Rap Anaphora Resolution Algorithm. In *proceedings of the Fourth International Conference on Language Resources and Evaluation.*

Rahman, A., & Ng, V. (2009). Supervised Models for Coreference Resolution. In *Proceedings of EMNLP* (pp. 968–977).

Resnik, P. (1995). Using Information Content to Evaluate Semantic Similarity in a Taxonomy. *arXiv preprint cmp-lg/9511007*.

Ringger, E., McClanahan, P., Haertel, R., Busby, G., Carmen, M., Carroll, J., ... Lonsdale, D. (2007). Active Learning for Part-of-speech Tagging: Accelerating Corpus Annotation. In *Proceedings of the Linguistic Annotation Workshop* (pp. 101–108).

Rus, V., Banjade, R., & Lintean, M. (2014). On Paraphrase Identification Corpora. In *Proceeding on the International Conference on Language Resources and Evaluation (LREC 2014).*

Rus, V., D'Mello, S., Hu, X., & Graesser, A. C. (2013). Recent Advances in

Conversational Intelligent Tutoring Systems. *AI Magazine*, *34*(3).

Rus, V., & Graesser, A. C. (2006). Deeper Natural Language Processing for Evaluating Student Answers in Intelligent Tutoring Systems. In *Proceedings of the National Conference on Artificial INtelligence* (Vol. 21, p. 1495).

Rus, V., & Graesser, A. C. (2009). The Question Generation Shared Task and Evaluation Challenge. In *The University of Memphis. National Science Foundation.*

Rus, V., & Lintean, M. (2012). A Comparison of Greedy and Optimal Assessment of Natural Language Student Input Using Word-to-word Similarity Metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP* (pp. 157–162).

Rus, V., Lintean, M., Banjade, R., Niraula, N., & Stefanescu, D. (2013). SEMILAR: The Semantic Similarity Toolkit. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.*

Rus, V., Lintean, M., Moldovan, C., Baggett, W., Niraula, N., & Morgan, B. (2012). The SIMILAR Corpus: A Resource to Foster the Qualitative Understanding of Semantic Similarity of Texts. In *Semantic Relations II: Enhancing Resources and Applications, The 8th Language Resources and Evaluation Conference (LREC 2012), May* (pp. 23–25).

Rus, V., Nan, X., Shiva, S. G., & Chen, Y. (2009). Clustering of Defect Reports Using Graph Partitioning Algorithms. In *SEKE* (pp. 442–445).

Rus, V., Stefanescu, D., Niraula, N., & Graesser, A. C. (2014). DeepTutor: Towards Macro- and Micro-Adaptive Conversational Intelligent Tutoring At Scale. In *Work in Progress Learning At Scale .*

Rus, V., Wyse, B., Piwek, P., Lintean, M., Stoyanchev, S., & Moldovan, C. (2010). The First Question Generation Shared Task Evaluation Challenge. In *Proceedings of the 6th International Natural Language Generation Conference*

(pp. 251–257).

Settles, B., & Craven, M. (2008). An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1070–1079).

Shen, D., Zhang, J., Su, J., Zhou, G., & Tan, C.-L. (2004). Multi-criteria-based Active Learning for Named Entity Recognition. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics* (p. 589).

Smith, A. K. S., Avinesh, P., & Kilgarriff, A. (2010). Gap-fill Tests for Language Learners: Corpus-Driven Item Generation. In *Proceedings of ICON-2010: 8th International Conference on Natural Language Processing.*

Soon, W. M., Ng, H. T., & Lim, D. C. Y. (2001). A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational linguistics*, *27*(4), pp. 521–544.

Stefanescu, D., Banjade, R., & Rus, V. (2014a). Latent Semantic Analysis Models on Wikipedia and TASA.

Stefanescu, D., Banjade, R., & Rus, V. (2014b). A Sentence Similarity Method Based on Chunking and Information Content. In *Computational Linguistics and Intelligent Text Processing* (pp. 442–453). Springer.

Stent, A. J., & Bangalore, S. (2010). Interaction Between Dialog Structure and Coreference Resolution. In *Spoken Language Technology Workshop (SLT), 2010 IEEE* (pp. 342–347).

Strube, M., & Müller, C. (2003). A Machine Learning Approach to Pronoun Resolution in Spoken Dialogue. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1* (pp. 168–175).

Sumita, E., Sugaya, F., & Yamamoto, S. (2005). Measuring Non-native Speakers' Proficiency of English By Using a Test with Automatically-generated Fill-in-the-blank Questions. In *Proceedings of the Second Workshop on*

Building Educational Applications Using NLP (pp. 61–68).

Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, *101*(476).

Turian, J., Ratinov, L., & Bengio, Y. (2010). Word Representations: A Simple and General Method for Semi-supervised Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (pp. 384–394).

Turney, P. D. (2002). Thumbs Up Or Thumbs Down?: Semantic Orientation Applied to Unsupervised Classification of Reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (pp. 417–424).

VanLehn, K. (2011a). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, *46*(4), pp. 197–221.

VanLehn, K. (2011b). The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educational Psychologist*, *46*(4), 197–221.

Varga, A., & Ha, L. A. (2010). Wlv: A Question Generation System for the Qgstec 2010 Task B. In *Proceedings of QG2010: The Third Workshop on Question Generation* (pp. 80–83).

Versley, Y., Ponzetto, S. P., Poesio, M., Eidelman, V., Jern, A., Smith, J., ... Moschitti, A. (2008). BART: A Modular Toolkit for Coreference Resolution. In *Proceedings of ACL* (pp. 9–12).

Wallach, H. M., Mimno, D. M., & McCallum, A. (2009). Rethinking LDA: Why Priors Matter. In *NIPS* (Vol. 22, pp. 1973–1981).

Williams, J. D., Niraula, N. B., Dasigi, P., Lakshmiratan, A., Suarez, C. G. J., Reddy, M., & Zweig, G. (2015). Rapidly Scaling Dialog Systems with Interactive Learning.

Wolfe, J. H. (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* (pp. 104–112). New York, NY, USA: ACM.

Wu, Z., & Palmer, M. (1994). Verbs Semantics and Lexical Selection. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics* (pp. 133–138).

Wyse, B., & Piwek, P. (2009). Generating Questions from Openlearn Study Units.

Yang, X., Su, J., & Tan, C. L. (2006). Kernel-based Pronoun Resolution with Structured Syntactic Knowledge. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics* (pp. 41–48).

Yao, X., & Zhang, Y. (2010). Question Generation with Minimal Recursion Semantics. In *Proceedings of QG2010: The Third Workshop on Question Generation* (p. 68).