1-1-2019

TOWARDS BUILDING INTELLIGENT COLLABORATIVE PROBLEM SOLVING SYSTEMS

Dipesh Gautam

TOWARDS BUILDING INTELLIGENT COLLABORATIVE PROBLEM SOLVING SYSTEMS

by

Dipesh Gautam

A Dissertation

Submitted in Partial Fulfilment of the

Requirements for the Degree of

Doctor of Philosophy

Major: Computer Science

The University of Memphis

August 2019

DEDICATION

To my parents.

ACKNOWLEDGMENTS

I express my sincere gratitude to my advisor Dr. Vasile Rus for his great support and guidance throughout the years of my PhD study. His enthusiasm in learning from students and participating in group discussion always motivated me to explore research problems and evolve myself as a researcher. I am also grateful to rest of the committee members Dr. Arthur C. Graesser, Dr. Deepak Venugopal and Dr. Scott D. Fleming for their valuable suggestions to shape my dissertation.

Furthermore, I thank my colleagues Nobal Niraula, Rajendra Banjade, Nabin Maharjan, Borhan Semai, Lasang Jimba Tamang and Nisrine Ait Khayi-Enyinda for participating in discussions and collaborations.

In addition, I am grateful to my parents, in-laws, brother, sister and relatives for their support and encouragements in every step of my life.

My special thank goes to my wife Tara for her love and care to me. It would not be possible to complete this long academic journey without her patience and support. Finally, I thank my son Sauvagya for his unconditional love. His innocent words always soothed me during hardships.

ABSTRACT

Dipesh Gautam. The University of Memphis. August, 2019. Towards Building Intelligent Collaborative Problem Solving Systems. Major Professor: Vasile Rus, Ph.D.

Historically, Collaborative Problem Solving (CPS) systems were more focused on Human Computer Interaction (HCI) issues, such as providing good experience of communication among the participants. Whereas, Intelligent Tutoring Systems (ITS) focus both on HCI issues as well as leveraging Artificial Intelligence (AI) techniques in their intelligent agents. This dissertation seeks to minimize the gap between CPS systems and ITS by adopting the methods used in ITS researches. To move towards this goal, we focus on analyzing interactions with textual inputs in online learning systems such as DeepTutor and Virtual Internships (VI) to understand their semantics and underlying intents.

In order to address the problem of assessing the student generated short text, this research explores firstly data driven machine learning models coupled with expert generated as well as general text analysis features. Secondly it explores method to utilize knowledge graph embedding for assessing student answer in ITS. Finally, it also explores a method using only standard reference examples generated by human teacher. Such method is useful when a new system has been deployed and no student data were available.

To handle negation in tutorial dialogue, this research explored a Long Short Term Memory (LSTM) based method. The advantage of this method is that it requires no human engineered features and performs comparably well with other models using human engineered features.

Another important analysis done in this research is to find speech acts in conversation utterances of multiple players in VI. Among various models, a noise label trained neural network model performed better in categorizing the speech acts of the utterances.

The learners' professional skill development in VI is characterized by the distribution of SKIVE elements, the components of epistemic frames. Inferring the population distribution of these elements could help to assess the learners' skill development. This research sought a Markov method to infer the population distribution of SKIVE elements, namely the stationary distribution of the elements.

While studying various aspects of interactions in our targeted learning systems, we motivate our research to replace the human mentor or tutor with intelligent agent. Introducing intelligent agent in place of human helps to reduce the cost as well as scale up the system.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table **Page**

Chapter 1

Introduction

1.1 Tutoring Systems and Collaborative Problem Solving Systems

Due to increased success and reduced cost in providing education with computer based learning systems, computer is becoming important tool for education. A study conducted by VanLehn (VanLehn, 2011) showed that the effectiveness of computer tutoring was as good as human tutoring. Moreover since the digital contents could be distributed very fast to a large number of learners with low cost, education providers are attracted towards using educational softwares.

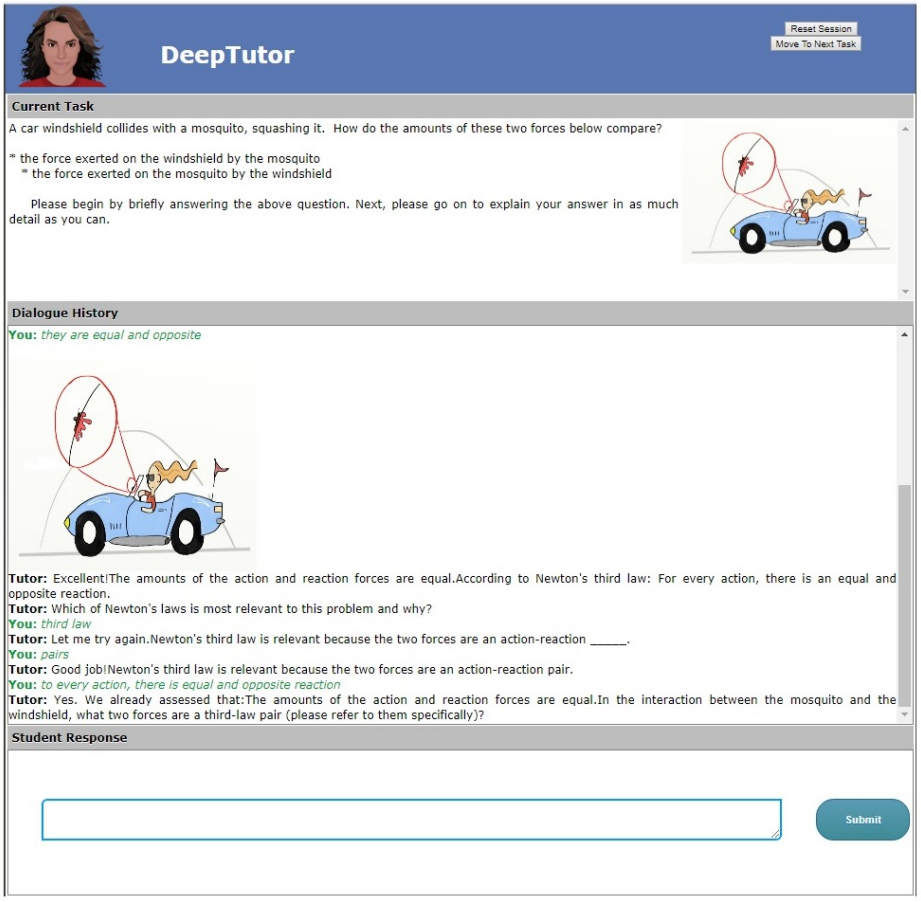

Fig. 1.1: DeepTutor interface

Various forms of learning systems ranging from one to one Intelligent Tutoring Systems (ITS) to multi-player Collaborative Problem Solving (CPS) Systems have been developed to the date. For instance, in one to one tutoring systems such as AutoTutor (Graesser, Lu, et al., 2004), DeepTutor (Rus, Stefanescu, Niraula, & Graesser, 2014) and Electronix Tutor (Graesser et al., 2018), students are asked different kinds of questions by the intelligent tutor and the students give answer. The students' answers are automatically assessed and then a feedback, hint or another question is generated depending upon the answer given by the student. The interface of DeepTutor (Figure 1.1) shows the problem description and the dialogue history. The problem, often referred as task, is authored by domain expert and consists of problem description, expected answers to the questions, prompts, possible hints and other relevant feedback. During tutoring, the system asks questions and the student respond to it in the form of natural language text and the dialogue continues. The dialogue consists of multiple cycle of tutor-student interaction in the form of (1)tutor question (2)student answer (3)relevant feedback from tutor until all the expectations of the task are complete after which the student are expected to master the concept.

Whereas in CPS systems such as Virtual Internships, students work in team on real world problems. They conduct background research, interview clients, develop and test prototypes and work with their colleagues to evaluate technical, social, economical and ethical impact and propose solution to the problem. In such system, the role of the mentor is to coordinate group discussions, provide feedback and assistance when students have questions, and assess proposal, submitted as free short text (notebook) by the students. An interface of Virtual Internship, for instance "Nephrotex" (Arastoopour et al., 2012), showing task, required sections of the task, a student submitted notebook (shown as a paragraph of text) and other details is shown in Figure 1.2. The participants in these Virtual Internships

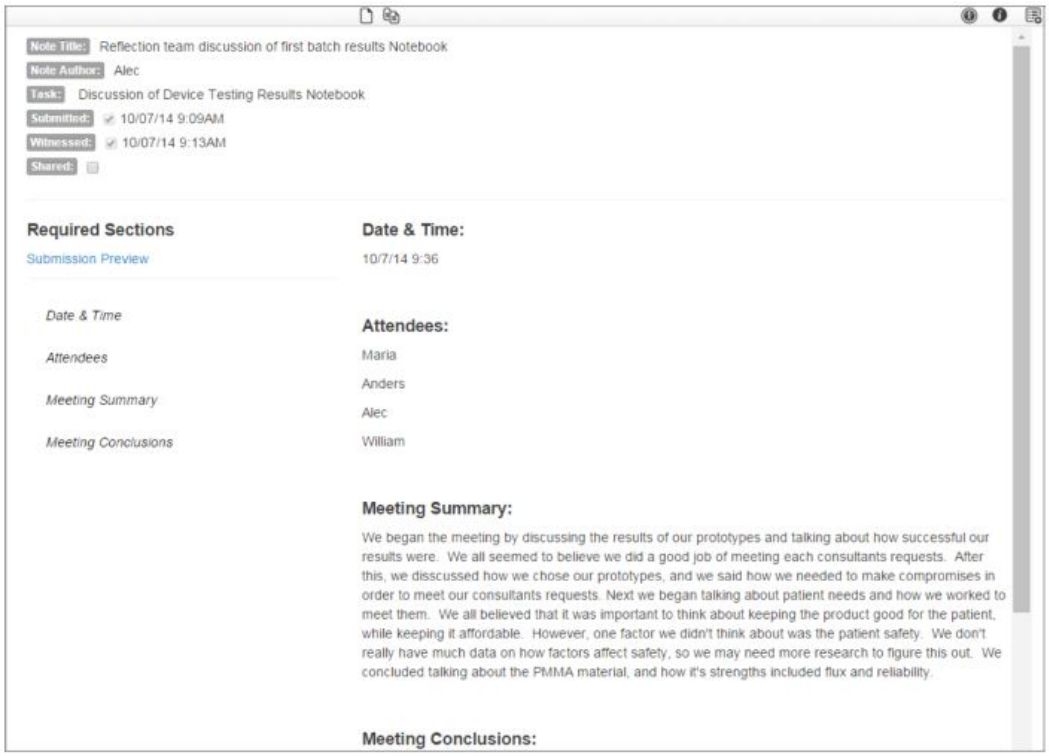

Fig. 1.2: Virtual internship interface showing student generated (notebook) entry

communicate via e-mail or by chat using the interface provided by the system. At present, mentoring is done by human mentors who have interest and knowledge on the project. Noting that the human involvement in assessment is a greater hindrance both in terms of cost and scalability, one of the objectives of our research is to automate the assessment process in order to achieve the bigger goal of replacing human mentors with an intelligent agent. Unlike one to one tutoring, the Virtual Internship is multi-player system where mentors intervene less during learning. However, the short answer assessment methods that were successful in tutoring systems could be adopted in assessing notebooks in Virtual Internships. Furthermore, the conversation utterances produced by the participants could be analyzed and used to improve the collaboration among the students.

1.2 Challenges and Approaches to Understand Students' Inputs

For the success of a learning system, not only the content but the quality of service that the system provides has a great impact. In one research (Wang & Chiu, 2011), it was noted that the interaction between learners though positively contributes to learning, the efficient and quality service to users has great impact on the success of the e-learning system.

Those qualities are characterized by the ability of the system to analyze the users' interactions efficiently with respect to time, cost and accuracy. While the users interact in various forms such as by multimedia, mouse click or by textual inputs, analyzing those interactions presents separate research challenges. We limit our research to explore methods of understanding natural language text inputs in ITS and CPS.

Table 1.1: An example showing reference answers, answer given by the student for a question asked by "*DeepTutor*". The student has to answer questions based on the problem description provided. "Q" refers to question, "SA" refers to student answer and "RA" refers to reference answer

Problem Description: A mover pushes a desk with a horizontal force such that the desk moves with constant velocity v0 across a carpeted floor. Suddenly, the mover doubles his force
Q:What can you say about the speed of the desk?
SA: *The velocity of the desk will increase but it will not necessarily double in velocity.*
RA1: *The desk moves with increasing velocity.* ***RA2:*** *The desk will move with constant acceleration and the speed will increase.* ***RA3:*** *The desk increases its speed as the net force is non-zero.* ***RA4:*** *The desk increases its speed as the net force is not zero anymore.*

While the natural language interactions could be the conversation, or free text writing, the type and length of interaction could vary widely from few words to few paragraphs. For example, the interactions in Intelligent Tutoring System occurs via textual conversational utterances whose length range from a single word to one or two sentences (Table 1.1). Those conversational texts entered by students are

expected to have targeted concepts that are needed to answer the questions asked to the student by the system.

On the other hand, in Virtual Internships, students submit a paragraph of text in the form of notebook as work proposal to the given problem (Table 1.2). In engineering design problem (Nephrotex and RescueShell; (Chesler et al., 2015)), these notebooks include design specifications such as material, manufacturing process, carbon nano-tube percentage, and the list goes on. Whereas in Land Science, a virtual internship of city zoning plan, notebooks include design plan which balances demands of stakeholders who advocate for indicators of community health. Moreover, students interact with their peers and mentors using chat or by e-mail.

Table 1.2: An example of notebook from the virtual internship "*Nephrotex*"

Design Specifications: The design has a CNT concentration of 20.0%, which is made through a dry-jet wet printing process. It uses a hydrophilic surfactant in PESPVP. Justification: The CNT concentration was chosen because it provides a disproportionately good performance for a linear increase in price. The dry-jet wet printing manufacturing process was chosen because it is the cheapest manufacturing option, which is an important factor for my client. I chose a hydrophilic surfactant because it is the most affordable surfactant, which makes up for its lack in marketability.

These text inputs could provide important information about the current state of the system, studens' learning, knowledge and skills, which could be used by learning systems to fulfill different purposes. To illustrate, in conversational systems such as ITS, the student's inputs are compared with ideal reference answers to check if student answer covered all or subset of expected concept or have misconception, and provide just in time feedbacks and hints. Also, discovering and classifying speech act of the utterances created by students is important in both ITS and CPS. For example, when a student says: *"Please give me an example of action and reaction force!"*, the system should be able to understand that the student is

requesting help, and able to provide a an instance of "action and reaction forces" that act in pair in real world.

Whereas in CPS systems such as virtual internships, the student generated notebooks are assessed by comparing with the rubrics or exemplars created by teachers. And ability to automatically assessing those notebooks reduces the cost and time. Specifically, at present, for every 30 students, at least 2 human graders are needed to assess notebooks in Virtual Internships. Replacing human assessor with computers not only scale up the system, but also improve the reliability by avoiding the bias induced by the traits of assessor.

Having said that the notebooks are assessed by comparing with rubrics, we explored two fundamental approaches. The first approach is data driven approach where the student notebooks are graded by human annotators based on rubrics and Machine Learning models are trained with these annotated notebooks. For that, the notebooks are converted into combination of features including domain expert features and general text analysis features inspired from previous work on automated essay scoring (Dikli, 2006; Leacock & Chodorow, 2003; Shermis & Burstein, 2003) and text analysis software tools such as CohMetrix (Graesser, McNamara, Louwerse, & Cai, 2004) and LIWC (Pennebaker, Francis, & Booth, 2001). The other approach uses only exemplars to generate core concepts classifiers. This approach is useful for a newly created Virtual Internships, where students' notebooks have not been collected yet to train the Machine Learning model. In this approach, the rubric is used only for final grading of notebooks after they are classified as presence or absence of core concepts. The concepts are keywords or phrases that are defined and tagged in few exemplars by teacher during authoring the Virtual Internship. These core concept classifiers use semantic similarity approach to find if core concept is present in some part of the text in student's notebook. The computation of such semantic similarity score could be based on

LSA or Neural Networks, both pre-trained with large domain general corpus. Our experiment found that the Neural Network performed better compared to LSA. However proper choice of threshold is a key to the success of semantic similarity approach. In contrast, the Regular Expression (RegEx) approach does not need threshold, however teacher has to generate keywords that could be used to create regular expression lists for each of the core concepts.

While semantic similarity based on LSA or Neural Network can perform better if expected concepts are mentioned directly in learners' responses. But it is difficult to assess in case those concepts are mentioned indirectly. For example, in DeepTutor, a target Intelligent Tutoring System (ITS) of our study, student could mention "*downward force from the earth*" instead of saying "*gravitational force*". In this case, even though both the concepts are same, the LSA based similarity score will be low. Such limitation of LSA based system could be avoided by using graph embedding method to learn better representation concepts and use them to assess student answer. Since such knowledge graphs encode relationship between the concepts in a huge graph, the indirect relation between concepts could be discovered easily from them.

Another important linguistic phenomenon in every human languages is the negation. Statements are negated implicitly (using words such as "avoid", "prevent", "prohibit") or explicitly (using cues such as "no", "not"). Presence of negation reverses the polarity of entire statements or of parts of statements. Many studies showed that negation accounts for a significantly large part of both spoken and written human language. In one study, it is reported that negation occurs twice as often in speech as in writing (Tottie, 1993). Some domain-specific corpus linguistics studies showed that negation occurs most frequently and represents a major portion of the information within such domain specific texts (Vincze, Szarvas, Farkas, Móra, & Csirik, 2008; Konstantinova et al., 2012). Also, an analysis of

student utterances in dialogues collected from experiments with actual students interacting with the intelligent tutoring system DeepTutor (Rus, D'Mello, Hu, & Graesser, 2013) showed that 9.36% of student utterances included explicit negation.

Negation presents different challenge in analyzing text to understand its semantics. Particularly, vector based semantic similarity using latent semantic analysis is not capable of identifying the the portion of the text negated. This results an adverse implication in the assessment of students' input in learning system by degrading the quality of feedback generated by the system and decreasing the reliability of overall assessment process and effectiveness of ITS. We address this issue by proposing a LSTM based negation handling approach in conversational system.

Besides assessment text response for their correctness, other kind of analysis such as speech act classification, and more specifically in Virtual Internships, assessment of professional skill development are needed to move a step closer towards developing intelligent agents in Learning Systems. The professional developments in systems like Virtual Internships are characterized by epistemic frame theory, which states that the learners develop the epistemic frames, or the network of skills, knowledge, identity, values and epistemology (SKIVE) that are unique to the professionals (Chesler, Bagley, Breckenfeld, West, & Shaffer, 2010). While communicating with participants in Virtual Internships, students activates different SKIVE components, whose distribution could be obtained by Markov Analysis using random walk. Such distribution could used to measure how much the students have developed professional skills necessary traits for the profession.

Moreover, speech act classification provides a method to understand conversation pattern among students (and possibly mentors or tutors) involved in conversation. Speech acts are a construct in linguistics and the philosophy of language that refers to the way natural language performs actions in

human-to-human language interactions, such as dialogues. Speech act theory was developed based on the "language as action" assumption. The basic idea is that behind every utterance there is an underlying speaker intent, called the speech act. For instance, the utterance "Hello, John!" corresponds to a greeting, that is, the speaker's intention is to greet, whereas the utterance "Which web browser are you using?" is about asking a question.

Labeling utterances with speech acts requires both an analysis of the utterance itself, e.g., "Hello" clearly indicates a greeting, but also accounting for the previous context, i.e., previous utterances in the conversation. For instance, after a question, a response most likely follows. This pattern holds in dialogues, i.e., interactions between two conversational partners, for instance student and tutor in DeepTutor, where there is a clear pattern of turn-taking; that is, a speaker's turn is followed by a turn by the other speaker. However, in multi-player conversations such as Virtual Internships, the one that we did our analysis, identifying the previous utterance that is most relevant to the current one is more difficult. In such multi-party conversation, the conversations are tangled and it becomes more challenging to link a target utterance to the previous one that triggered it. The complexity of untangling such multi-player conversations is further increased as the number of participants increases.

1.3 Research Objectives

While the bigger goal of this dissertation is to automate the assessment of learners' input and develop an intelligent agent, the specific objectives are listed as follows:

1. Assessment of short text content
2. Concept classifier generation from reference examples
3. LSTM based negation handling

4. Markov analysis of Learners' conversation in multi-player system

5. Speech act categorization in multi-player system

6. Use knowledge graph embedding to assess students' answer in intelligent tutoring system

1.4 Contributions

This dissertation makes following contributions towards understanding interaction in learning systems and automating the assessment of learners' generated textual contents.

In Chapter 2 we present a machine learning approach to automatically assess student generated textual design justifications in engineering virtual internships. Particularly we compare two major categories of models coupled with machine algorithms: domain expert-driven vs. general text analysis models. We found no quantitative differences among the two major categories of models, domain expert-driven vs. general text analysis, although there are major qualitative differences as discussed in the paper.

In Chapter 3 we present a method for generating classifiers using specifications provided by teachers during their authoring process instead of participant data. Our models rely on Latent Semantic Analysis based and Neural Network based semantic similarity approaches in which notebook entries are compared to ideal, expert generated responses. Furthermore, in Chapter 4 we report on our effort to develop an LSA-based assessment method without student data. we also investigate the optimum corpus size and vector dimensionality for these LSA-based methods.

In Chapter 5 we present a novel approach to automatically handling negation in tutorial dialogues using deep learning methods. Specifically, we explored various Long Short Term Memory (LSTM) models to automatically detect negation focus,

scope and cue in tutorial dialogues collected from experiments with actual students interacting with the intelligent tutoring system, DeepTutor.

In Chapter 6 we conduct a Markov analysis of learners' professional skill development based on their conversations in virtual internships, an emerging category of learning systems characterized by the epistemic frame theory. We model individual students' development of epistemic frames as Markov processes and infer the stationary distribution of this process, i.e. of the SKIVE elements.

In Chapter 7 we present a novel approach to classify speech acts of chat utterances in mylti-player system such as virtual internships, where there is no clear turn taking among the speakers. Our approach is based on pre-training a neural network on a large set of noisy labeled data.

In Chapter 8 we present a method to use knowledge graph for assessing student answers in dialogue based intelligent tutoring systems. We focus to develop and evaluate our model for DeepTutor, an intelligent tutoring system for high school physics students.

We conclude with Conclusions and Future Works in Chapter 9.

Chapter 2

Assessment of Short Text Content

In Engineering Virtual Internships, students work as interns at fictional companies to create engineering designs. To improve the scalability of these virtual internships, a reliable automated assessment system for tasks submitted by students is necessary. Therefore, we propose a machine learning approach to automatically assess student generated textual design justifications in two engineering virtual internships, Nephrotex and RescuShell. To this end, we compared two major categories of models: domain expert-driven vs. general text analysis models. The models were coupled with machine learning algorithms and evaluated using 10-fold cross validation. We found no quantitative differences among the two major categories of models, domain expert-driven vs. general text analysis, although there are major qualitative differences as discussed in this chapter.

2.1 Introduction

In virtual internships, students play the role of interns in a virtual training environment. In engineering virtual internships, such as Nephrotex (NTX) and RescueShell (RS), students research and create multiple engineering designs (Chesler et al., 2015). As part of their design process, they regularly submit written work in the form of electronic engineering notebooks that are assessed by human judges. This human assessment is labor intensive, time consuming, and error-prone under certain circumstances such as time pressure. Furthermore, prior work has suggested that the reliability of human assessments can vary depending on the traits of the assessor, their experience, and the types of problems being assessed (Tisi, Whitehouse, Maughan, & Burdett, 2013) . Thus, an automated assessment method that could provide efficiency in terms of time and cost as well as improved reliability is much needed. Our work presented here constitutes a step in this direction.

In the present study, we explored various models for automatically assessing notebooks in the engineering virtual internships NTX and RS. The content of these notebooks varies; however, in this study we focus on only one type of notebook in which students must justify their engineering designs by typing a short, free-text justification. We have experimented with models that emulate an expert analysis of the student notebook entries as well as models derived from general textual analysis features. It should be noted that our work differs from previous attempts which rely on a semantic similarity approach, i.e. measuring how semantically close a student-generated response is to an ideal, expert-generated response as in (Mohler & Mihalcea, 2009).

The domain expert-driven models incorporate theoretically driven, content-based features identified by human experts such as "referencing any performance parameter such as cost", which is a general design feature because it applies to all engineering designs in NTX and RS, or "indicating the power source", a feature specific to the concrete task of designing an exoskeleton, which was the focus of the RS internship and not NTX. A challenge with the domain expert-driven models is that the features are specific to either the type of task, e.g. engineering design, or the concrete task itself, e.g. design an exoskeleton. This results in a scalability issue as these models must be redesigned manually by domain experts when moving to a new domain, new type of task, and/or a new concrete task. However, the net theoretical advantage of these domain expert-driven models is that they are tailored to the task at hand and therefore are expected to yield very good performance. These models also afford the ability to create automatic and tailored feedback to students given their task-specific diagnostic capabilities.

The other category of models that we used rely on general text analysis features inspired from previous work on automated essay scoring (Dikli, 2006; Leacock & Chodorow, 2003; Shermis & Burstein, 2003) and text analysis software

tools such as CohMetrix (Graesser, McNamara, et al., 2004)and LIWC (Pennebaker et al., 2001). For instance, in automated essay scoring the length in words of the essay, i.e. the number of all word occurrences or word tokens, is by far the best predictor of essay quality. Coh-Metrix is a software package that calculates the coherence of texts in terms of co-reference, temporal cohesion, spatial cohesion, structural cohesion, and causal/intentional cohesion. LIWC (Linguistic Inquiry and Word Count) uses a word count strategy to characterize texts along a number of dimensions that include standard language categories (e.g., articles, prepositions, pronouns), psychological processes (e.g., positive and negative emotion word categories), and traditional content dimensions (e.g., sex, death, home, occupation).

The key advantage of the general text analysis models is that they are generally applicable across types of tasks, specific tasks, and domains. In addition, the general text analysis features are relatively cost-effective and easy to derive from the data compared to features derived by domain experts, which require (significantly more) human time and effort.

In this work, we explore the predictive power of the two major categories of models mentioned above, domain-expert vs. general text analysis, in conjunction with a number of machine learning algorithms such as decision trees, naïve Bayes, Bayes Nets, and logistic regression. Furthermore, we employed an ensemble of classifiers approach in order to boost the performance of individual models. We conclude the chapter with a qualitative assessment of the relative benefits of the proposed models for virtual internships by considering their predictive value, the labor involved in their development, and their ability to provide interpretable assessments for students.

2.2 Background

We review in this section prior work on assessing students' openended responses with an emphasis on prior work in the area of educational technologies.

Automated essay scoring systems (Dikli, 2006; Leacock & Chodorow, 2003; Shermis & Burstein, 2003) have been developed for more than two decades as a way to tackle the costs, reliability, generality, and scalability challenges associated with assessing student generated open-ended responses to essay prompts. There are a number of systems available for automated essay scoring, some of which are commercial. It is beyond the scope of this dissertation to offer a thorough review of the work in this area. We limit ourselves to noting that the focus on automated essay scoring is on the argumentative power of an entire essay while in our case the focus is on required (design) items that must be present in paragraph-like justifications. This entails that style and higher level constructs such as rhetorical structure are less important in our task as opposed to the essay scoring task and that factors that focus more on content measures are highly important. Given these differences and the fact that the two most predictive factors of essay quality are also content related, we included in our models the following two features: word count, i.e. total number of word occurrences or tokens in student justifications, and content word count, i.e. the total number of content word occurrences (nouns, verbs, adjectives, and adverbs).

Directly relevant to our study is previous work by Rus, Feng, Brandon, Crossley, and McNamara (Rus, Feng, Brandon, Crossley, & McNamara, 2011) who studied the problem of assessing student-generated paraphrases in the context of a writing strategy training tutoring system. One of the strategies in this tutoring system is paraphrasing. As the system is supposed to prompt students to paraphrase and then provide feedback on their paraphrases, Rus and colleagues collected a large corpus of student-generated paraphrases and analyzed them along several dozen linguistic dimensions ranging from cohesion to lexical diversity obtained from Coh-Metrix (Graesser, McNamara, et al., 2004). There are significant differences between their work and ours. First, we deal with justifications which can

vary in length from a few words to a full paragraph as opposed to explicitly elicited paraphrases of target sentences. Second, we do use extra features to build our models besides the Coh-Metrix indices. Third, we assess the student generated justifications as acceptable or unacceptable (i.e., correct or incorrect). We could eventually investigate finer levels of correctness, e.g. on a scale from 1-5, which we plan to do as part of our future work.

Williams and D'Mello (Williams & D'Mello, 2010)worked on predicting the quality of student answers (as error-ridden, vague, partially-correct or correct) to human tutor questions, based on dictionary-based dialogue features previously shown to be good detectors of cognitive processes (Williams & D'Mello, 2010, cf.). To extract these features, they used LIWC (Linguistic Inquiry and Word Count; (Pennebaker et al., 2001)), a text analysis software program that calculates the degree to which people use various categories of words across a wide array of texts genres. They reported that pronouns (e.g. I, they, those) and discrepant terms (e.g. should, could, would) are good predictors of the conceptual quality of student responses. Like Williams and D'Mello, we do use LIWC to analyze student notebooks' justifications. Furthermore, we employ expert-identified features and features from Coh-Metrix and automated essay scoring.

Prior work by Rus, Lintean, and Azevedo (Rus, Lintean, & Azevedo, 2009)investigated the performance of several automated models designed to infer the mental models of students participating in an intelligent tutoring system (ITS). The ITS was designed to teach students self regulatory processes while they were learning about science topics such as the human circulatory system. Rus and colleagues used two methods, a content-based method and a word-weighting method, to derive features for their models. While our present work does not investigate models using word-weighting methods, we do investigate models using content-based features.

The content-based features used by Rus and colleagues included a taxonomy of relevant biology concepts derived by human experts, expert annotated pages of content from the ITS, and expert generated paragraphs. In the present study, the content-based features, or domain-expert (DE) features, we used consist of discourse codes developed by human experts. Discourse codes indicate the presence or absence of specific concepts in student talk, or in this case, student written work. The DE features were developed through a grounded analysis of student design justifications collected from engineering virtual internships (Glaser & Strauss, 2009).

The learning that occurs in engineering virtual internships can be characterized by epistemic frame theory. This theory claims that professionals develop epistemic frames, or the network of skills, knowledge, identity, values, and epistemology that are unique to that profession (D. W. Shaffer, 2006b). For example, engineers share ways of understanding and doing (knowledge and skills); beliefs about which problems are worth investigating (values), characteristics that define them as members of the profession (identity), and a ways of justifying decisions (epistemology). In this study, we used epistemic frame theory to guide the development of the DE features. In prior work, elements of the engineering epistemic frame have been operationalized as discourse codes and used to assess engineering thinking in virtual internships (Chesler et al., 2015). In this study, the DE features we identified correspond to elements of the engineering epistemic frame that relate to justifying design decisions. The presence or absence of these features in a student's written work thus represents elements of the engineering epistemic frame that are present or lacking.

In sum, we used some of the features described by the above researchers in our work, such as word count, as well as novel features, e.g. features based on the engineering epistemic frame.

2.3 Engineering Virtual Internships

In this study, we examined student written work collected from the engineering virtual internships, Nephrotex (NTX) and RescueShell (RS). In NTX, students work in teams to design filtration membranes for hemodialysis machines, while in RS, student teams design the legs of a mechanical exoskeleton used by rescue workers.

All interactions in virtual internships take place via a website in which students communicate with their teams using email and chat. During the internships, students research and create engineering designs in two cycles. In each cycle, students design five prototypes and later receive performance results for each prototype which they have to analyze and interpret.

During their design process, students submit records of their work via electronic notebook entries for each substantive task they complete, including summarizing research reports and justifying design decisions. The expectations of notebook entries are outlined in prompts, which students receive via email in the virtual internship website. Each notebook that students submit is divided into notebook sections, i.e., separate text fields for items that are defined by the email prompts. In this study, we analyzed notebook sections in which students provided justifications for their prototype design decisions.

Once students complete each notebook section, they submit the notebooks to trained human raters for assessment. In the fiction of the virtual internships, these raters play the role of more senior employees in the company who act as mentors to the students. The role of the mentors is to answer student questions and lead team discussions, in addition to assessing student work.

Once a mentor receives a notebook, they assess each section as acceptable or unacceptable using provided rubrics. The assessment system used by the mentors automatically generates pre-scripted feedback corresponding to the assessment given

to each section. Currently, this feedback is generic in the sense that it does not respond to the particulars of a student's response. For example, an assessment of unacceptable on a notebook section requiring a summary generates feedback that (1) informs the student that the section was unacceptable, (2) reminds them of the content they were asked to summarize, and (3) points them to the documents they were asked to summarize. This automated feedback does not inform the student exactly why the section was rated as unacceptable. However, the mentor does have the option to compose specific feedback for the student if they wish.

Our work here moves us towards a more automated and student tailored assessment and feedback mechanisms which could have significant impact on the economy of scaling virtual internships to all students, anytime, anywhere via Internet-connected devices.

2.4 Experiments and Results

We describe first the data set we used in our experiments before presenting the experiments and results obtained with the models.

2.4.1 Dataset

In this study, we analyzed notebook sections from the NTX and RS virtual internships in which students justified their engineering design decisions. In these notebook sections, students were required to include the design input choices they selected—that is, their design specifications, and a justification explaining why this design was chosen for testing.

Mentors assessed these notebook entries as acceptable or unacceptable in real-time during the virtual internship using the following rubric:

1. Listed their design specifications
2. Included a justification referencing at least one design specification.

Acceptable justification may include:

1. Prioritizing attributes
2. Referencing internal consultant requests
3. The performance of a design specification on a specific attribute
4. Experimental justifications (e.g., holding design specifications constant)

To select data for this study, we randomly sampled 298 justification sections from 20 virtual internship sites, i.e. datasets corresponding to 20 schools where the virtual internships were implemented. Twelve were NTX sites and eight were RS sites. Of the 298 justifications sampled, 146 were from NTX and 152 were from RS. Students were given the same prompts for justification sections in NTX and RS. In addition, the same rubrics were used by raters in NTX and RS. Thus, we combined data from RS and NTX to train our models.

As described above, justification sections were originally assessed by mentors during the virtual internship in real time. The mentors were trained to assess notebook section, but they were not experts in the domain of engineering or the content of the virtual internships. In addition, they had to assess notebook sections under time constraints and while completing their other responsibilities as a mentor. For example, they could have to respond to student questions via chat while assessing. Thus, to obtain potentially more valid and reliable assessments for model training, the justification sections in this study were re-assessed by more experienced raters that did not face the constraints placed on the mentors. We found that the agreement between the human mentors and our experienced raters on the 298 student justifications we used in this work was kappa = 0.271. This value is very low, indicating that mentors' assessments are not reliable, as we suspected.

Each justification section was re-assessed by two new raters, benchmark rater 1 (BE1) and benchmark rater 2 (BE2). BE1 had over two years of experience rating notebook sections from virtual internships and had contributed to the content

Table 2.1: Example of acceptable and unacceptable notebooks from the virtual internship "*Nephrotex*"

Notebook entry	Assessment
Design Specifications: PAM, Vapor, Negative Charge, 4 % Justification: This prototype was altered slightly from the original with this material by changing from 2% CNT to 4%. This is an attempt to increase reliability without hindering flux or blood cell reactivity.	Acceptable
Design Specifications: PAM, Vapor, Negative Charge, 2.0 Justification: These specificaions ran best for PAM material	Unacceptable

Table 2.2: Distribution of human-ratings in the 298 instances

Human Rating	#Instances
Acceptable	217
Unacceptable	81
Total	298

development of both NTX and RS. BE1 was thus considered an expert rater for the purposes of this study. BE2 was a less experienced rater trained to assess justification sections. BE1 and BE2 assessed all 298 justification sections using the rubric above and agreed on one final judgement (acceptable or unacceptable) for each justification. Their inter-annotator reliability as measured by kappa was 0.767. Table 2.1 includes examples of notebook sections from NTX assessed as acceptable and unacceptable by the benchmark raters. About 73% of the instances in the data set were rated positively by the BEs. The distribution of positive and negative instances is shown in Table 2.2.

2.4.2 Feature Selection

As already mentioned, we focused on two major categories of models: models that rely on domain-experts (DE) versus models that rely on more general textual analysis features. We developed the DE features through a grounded analysis (Glaser & Strauss, 2009)of a sample of 98 justification sections. These features were developed by two researchers who re-assessed the sample and developed discourse codes corresponding to what they attended to while assessing. Next, we automated these codes using the nCoder, a tool for developing and validating automated

discourse codes that relies on authoring targeted regular expressions for each of the expert identified codes (D. Shaffer et al., 2015). These codes were included as features in our models (see Table 2.3 for descriptions).

The general textual analysis features were further divided by their source into the following three categories: features inspired from automated essay scoring (ES) research, features obtained with the automated tool for textual analysis Coh-Metrix, and features obtained with the automated tool for textual analysis LIWC. This categorization of the general textual analysis features is needed for several reasons. First, the various sources capture different aspects of a text. Second, this categorization allows us to conduct ablation studies in which we assess the contribution of each major category of features to solving the task at hand. It should be noted that there is overlap among the features from various groups/sources. For instance, the WC (LIWC), DESWC (Coh-Metrix), and Word_Count (DE) features are all counts of white-spaces in a target text, i.e. justifications in our case. These features are slightly different from the token Count feature in the ES group which counts number of tokens after applying the Stanford tokenizer tool. Similar features will not end up in the same models if they correlate highly, as explained next.

Not all features have equal predictive power and having redundant or irrelevant features can decrease the performance of the models. Therefore, we had a feature selection step keeping features that have low correlation with each other ($<.70$). When two features in a model had a correlation greater than .70 of them was dropped. For instance, from the LIWC and Coh-Metrix groups of features the features selected via this process were: WC, SIXLTR, adverbs, verbs, DESSC, DESSL, DESSLd, PCNARz, PCCONNp (See Table 2.3 for descriptions). The feature selection step was needed given that we worked with various machine

Table 2.3: Descriptions of the some features used in the proposed models (not all shown due to space constraints)

Features	Description
	LIWC(LI)
Word Count	Word Count-(WC: Total number of words in text); *Token Count*-(TC: Number of unique words in text); *Words >6 letters*- (SIXLTR: total number of words greater than 6 letters); *Punctuations*
Type Token Ratio	*Ratio of TC and WC*
	Coh-Metrix(CM)
Lexical Component Counts	*DESPC* - (Paragraph count: number of paragraphs); *DESSC* - (Sentence count: number of sentences); *DESWC* - (Word count: number of words)
DESPL	DESPL - (Paragraph length: number of sentences, mean); *DESPLd* - (Paragraph length, standard deviation); *DESSLd*- (Sentence length, standard deviation)
Connectives Features	*PCCONNp* - (the degree to which the text contains connectives such as adversative, additives and comparative connectives to express relations in the text.)
Temporality Features	*PCTEMPz* - (the temporality such as tense or aspect of the text); *SMTEMP* - (temporal cohesion, measured by repetition score of tense and aspect)
LDTTRa	Type token ratio of all words.
	Domain Expert(DE)
Exoskeleton Design Inputs	Control Sensor, Range of Motion, Power Source, Material, Actuator
Dialyzer Design Inputs	Process, Surfactant, Material, Carbon Nanotube Percentage
Attributes	Referencing any design attribute or performance parameter such as cost, reliability, etc.
Justification Features	*Balancing* - Justifying input choices by stating it made up for the weakness of another choice or by saying that another choice will balance out its weaknesses; *Client* - Justifying input choices by stating it would be good for the client or end user of the product; *Consultant Requests* - Justifying input choices because the results meet or are expected to meet internal consultants' requests; *Evaluation* - Justifying input choices by evaluating the performance of the inputs
	Essay Scoring (ES)
Token Count	Count of word occurrences in the justification.
Content Word Count	Count of all content words (noun, adjective, verb, adverb) in the justification.

learning algorithms, some of which do not have a feature selection process linked to them, e.g. the stepwise variable selection in some regression implementations.

2.5 Results

We experimented with the proposed models in conjunction with a number of classification algorithms including decision trees, naïve Bayes, Bayes Nets, and logistic regression. We present here the results obtained with the logistic regression classifier as it yielded the best results overall. The models were validated using 10-fold cross validation. Performance was measured using standard measures such as accuracy, false positive rate, precision, recall, F measure, and kappa statistic. The false positive rate, the percentage of true negatives predicted as positives, is of special interest because it gives us an idea of how many justifications are deemed correct when in fact are not, by a particular method. That is, it indicates how many opportunities for feedback a specific method might miss as a justification deemed correct means there is no need for specific feedback to improve it. The evaluation results are shown in Table 2.4. We focus next on the most important model comparisons due to space constraints, e.g. we do not show results when combining two groups of features.

Table 2.4: Performance evaluation results for various models

S.N.	Features	Acc	FPR	P	R	F1	Kappa
1	ES	85.2349	0.249	0.85	0.852	0.851	0.6181
2	LI	83.2215	0.295	0.827	0.832	0.829	0.5591
3	CM	85.2349	0.295	0.848	0.852	0.846	0.5991
4	DE	83.2215	0.302	0.827	0.832	0.828	0.5555
5	ES + LI	84.8993	0.258	0.846	0.849	0.847	0.6079
6	ES + CM	83.557	0.301	0.83	0.836	0.831	0.5626
7	ES + DE	83.8926	0.277	0.835	0.839	0.836	0.5801
8	LI + CM	82.5503	0.328	0.818	0.826	0.819	0.5301
9	LI + DE	81.5436	0.293	0.813	0.815	0.814	0.5283
10	DE + CM	82.8859	0.319	0.822	0.829	0.823	0.541
11	ES+LI+CM	83.8926	0.292	0.834	0.839	0.835	0.5733
12	LI + DE + CM + ES	81.8792	0.3	0.815	0.819	0.817	0.5314

Table 2.5: Classification performance of ensemble models with majority voting (rows 1 and 2 include weakest individual and combined models, respectively; rows 3 and 4 include the best individual and combined models, respectively)

S.N.	Features	Acc	FPR	P	R	F1	Kappa
1	(DE)+(LI)+(ES)	0.8523	0.264	0.8483	0.8523	0.8503	0.612
2	(LI_+DE)+ (LI+DE+CM+ES)+(LI+ CM)	0.8624	0.237	0.8595	0.8624	0.8609	0.6427
3	(ES) + (CM) + (DE)	0.8658	0.2435	0.8624	0.8658	0.8641	0.6473
4	(ES+LI)+(ES+DE)+ (ES+CM)	0.8557	0.2627	0.8516	0.8557	0.8536	0.6192

We started with models that included features from only one group, i.e. the individual feature group models shown in rows 1-4 in Table 2.4, selected the best such model and then added, sequentially, features from the other groups in batches, where each batch contained the selected features in one group. This procedure, also known as an ablation study in machine learning, allows to see what we gain if we add a group of features to a model that already contains feature from one or more groups. From Table 2.4, we infer that the ES and Coh-Metrix individual models are the best as they have slightly higher accuracy in prediction (85.23% for ES and 85.23% for Coh-Metrix) compared to other two individual feature groups. Also their kappas are the highest among the models with only one group of features.

In row 5, we show the results when combining all general text analysis features: ES, LIWC, and Coh-Metrix. As already mentioned before, we are directly interested in comparing the domain expert-driven model, derived from the DE features, with the model in row 5 that includes all the general text analysis features from the ES, LIWC, and Coh-Metrix groups. As we notice, these two qualitatively different models have very similar performance across all performance measures.

In addition to developing the above models from subsets of features, we used ensembles of 3 individual and combined models, respectively, in conjunction with a majority voting mechanism. For instance, if 2 or 3 out of 3 models predicted a justification as accepted then the final prediction for the instance was accepted. We

experimented with voting in two different ways: (1) we used the best 3 models from the individual or combined groups of features; (2) we used the weakest 3 models obtained with any combinations of features from individual and combined groups of features; this latter case is based on results from statistics that show that combining weak classifiers should result, in general, in better performance relative to the performance of each of the weak classifiers. Both types of ensembles (weakest versus best) yielded in the best cases similar accuracies of 86% and similar performance across all the other performance measures (see Table 2.5). The false positive rate of the weakest combined model ensemble was lowest.

2.6 Conclusions

In this work, we experimented with multiple models designed to automatically assess notebook sections from engineering virtual internships. In particular, we developed models to assess notebook sections in which students justified design decisions. All models performed very well with good and very good kappa scores (kappas scores of 0.6-0.8 are considered very good) indicating that they are much better than chance predictions. Our results show that, in this context, the predictive value of models using only the general text analysis features is comparable to the predictive value of a model using only the DE features (a McNemar's test on paired nominal data revealed no significant difference between the two models' prediction).

In particular, the ES group of features is the best predictor of students' justifications quality. When other groups of features are added to the individual ES model, the results do not improve significantly. The fact that the ES features are so good is not surprising. Word count, or essay length, which is one of the features in the ES group, is known as being the best predictor of essay quality in automated essay grading (Mohler & Mihalcea, 2009; Rus & Niraula, 2012). Also, the CohMetrix group of features are a good predictor of the quality of students' justifications.

It is important to note, however, that the predictive power of a model is only one dimension for evaluating the utility of automated assessment models in learning environments like virtual internships. We suggest that developmental cost and interpretability of the models are also valuable dimensions to consider. Of the models presented above, those using only the general text analysis features have the lowest developmental cost. Moreover, these features are generally applicable across types of tasks, specific tasks, and domains. In contrast, models containing the DE features have a relatively high developmental cost because their features required the time and expertise of humans to develop. We do note that the DE features described in this chapter were automated. Thus, they can readily be applied to more justification sections from engineering virtual internships. However, these DE features are specific to this context and are likely not generalizable outside of engineering virtual internships.

The utility of these automated assessment models lies in implementing them in real-time during a virtual internship where they will be used to assess student work and either generate automatic feedback or suggest feedback for human mentors to give. For the models using only the general text analysis features, any potential feedback would be in terms of features such as word count or "narrativity" of the text that are not directly related to the domain-relevant content of the text. Those models using DE features, however, could potentially generate domain-relevant feedback in terms of what DE features were present and absent in the text. For example, if a student's justification section fails to relate their design decisions to the requests of the company's internal consultants, that is, it lacks the "Consultant Requests" DE feature, feedback could be suggested to the mentor or provided automatically to the student informing them of this missing information and suggesting ways to include it. Thus, in terms of ease of interpretation, those

models using only the general text analysis features have a relatively low ease of interpretation compared to those models that include the DE features.

In this context, we then suggest the use of the best predictive model to assess the overall quality of justifications in engineering virtual engineering internships, and subsequently use the DEbased model to identify potential domain-specific missing parts in an unacceptable justification in order to provide direct feedback to the student or at least make suggestions to human mentors regarding possible weak aspects of the justification. This approach balances the tradeoffs between generality and reliability versus domain and task specific diagnostic capabilities.

We plan to further improve the predictive power, generality, and diagnostic capabilities of our models. For instance, we are considering unsupervised methods to automatically detect domain specific codes that could be used as features in our DE models. Furthermore, we are considering unsupervised topic detection in student-generated justification as a way to generalize the applicability of our models to other domains and types of tasks.

Chapter 3

Concept Classifier Generation from Reference Examples

While working as interns in virtual internships, participants complete activities and then submit write-ups in the form of short answers, digital notebook entries. Unlike prior work that used classifiers trained on participant data to automatically assess notebook entries, we evaluate a method for generating classifiers using specifications provided by teachers during their authoring process. Our models rely on Latent Semantic Analysis based and Neural Network based semantic similarity approaches in which notebook entries are compared to ideal, expert generated responses. We also investigated a Regular Expression based model. The experiments on the proposed models on unseen data showed high precision and recall values for some classifiers using a similarity based approach. Regular Expression based classifiers performed better where the other two approaches did not, suggesting that these approaches may complement one another in future work.

3.1 Introduction

Recently, authoring tools have been developed that let teachers customize and create new versions of digital learning environments such as intelligent tutoring systems and simulations (D. W. Shaffer, Ruis, & Graesser, 2015). However, if these environments use integrated automated systems, such as classifiers, customization can be problematic: a new environment invalidates previous automated systems and participant data does not yet exist to train new ones. Therefore, teachers who author these learning environments must implement them, at least initially, without a key component of the technology.

For example, during virtual internships, participants complete activities and submit work in the form of digital notebook entries. Typically, these are short answer responses ranging from a few sentences to a paragraph in length. Prior work

has investigated automated assessment of notebook entries by training classifiers on participant data (Rus, Gautam, Swiecki, Shaffer, & Graesser, 2016). However, since the development of the Virtual Internship Authoring Tool (Swiecki & Shaffer, 2017), teachers can now customize activities and their notebook requirements. Thus, previously developed classifiers may no longer be valid and, initially, participant data is not available to use for model training.

In this work, we present and test a method that addresses this issue by generating classifiers from specifications that teachers provide during the authoring process rather than waiting to generate them from participant data. Ultimately, these classifiers will be integrated into a fully automated assessment system that will score participant notebook entries. In this study, however, we only report on the development of classifiers for determining whether teacher defined requirements are present or absent in an entry, not classifiers that assign a final assessment.

3.2 Background

Several automated essay scoring systems (Dikli, 2006; Leacock & Chodorow, 2003; Shermis & Burstein, 2003)have been developed to tackle the challenges of costs, reliability, generality and scalability while assessing open-ended essays. Previous researches on automated essay scoring focused on the argumentative power of an entire essay, while in our case, the student generated content is typically short text the length of a sentence or paragraph. Also, the focus of our assessment is to classify the content based on the presence or absence of semantic content defined by teachers during their authoring process. This means that style and higher-level constructs, such as rhetorical structure, are less important in our task compared to essay scoring and that factors that focus more on content measures are more important. Therefore, we limit our work to a semantic similarity approach and Regular Expression (RegEx) matching approach to identify the presence of targeted semantic content in participant generated text.

Various methods of text similarity measures have been used from the very early years of information retrieval. One of the simplest approach is to use the lexical overlap between the texts, however this approach does not consider the semantic relation between the words. Salton & Lesk (Salton & Lesk, 1968)used is term frequency based vector model for documents similarity. Such model fails when two texts with same meaning have few overlapping words. Other approaches use knowledge base such as WordNet to find semantically similar words in two text (Fernando & Stevenson, 2008; Lintean & Rus, 2012). However these approaches face challenges of word sense disambiguation. Other approaches use LSA or LDA methods that rely on large corpus and do not face word sense disambiguation challenge (Rus, Lintean, Graesser, & McNamara, 2009).

Rus et al. (Rus, Lintean, Graesser, & McNamara, 2009)collected a large corpus of student-generated paraphrases and analyzed them along several dozen linguistic dimensions ranging from cohesion to lexical diversity obtained from Coh-Metrix (Graesser, McNamara, et al., 2004). They used the most significant indices to build a prediction model that can identify true and false paraphrases and also several categories of paraphrase types. Our work is significantly different than their work as our classifier model does not rely on participant generated content (we develop classifiers from teachers specifications of content before any participant response is available), secondly our paraphrase detection model measures semantic relation between the text without depending on linguistic features such as content word counts.

Our LSA based similarity method relies on the combination of constituent words a phrase. Hence the similarity score will be more biased towards phrases having common words. While the Neural Network (NN) based semantic similarity method proposed by (P.-S. Huang et al., 2013; Shen, He, Gao, Deng, & Mesnil, 2014), which we also explored, projects the phrase pairs into common low

dimensional space hence the similarity score obtained will be more consistent irrespective of the presence of common words in the phrases.

Our work closely relies on previous works (Corley & Mihalcea, 2005; Fernando & Stevenson, 2008; Lintean & Rus, 2012)where the authors proposed methods to measure the semantic similarity between texts. The authors in (Corley & Mihalcea, 2005)and (Fernando & Stevenson, 2008)used knowledge bases such as WordNet while the authors in (Lintean & Rus, 2012)used word to word similarity and vectorial representation of words derived using Latent Semantic Analysis (LSA) to compute the semantic similarity of two given texts. In addition to these methods, we used in our work presented here phrase vectors generated using Neural Network based models (P.-S. Huang et al., 2013; Shen et al., 2014).

Our work is also partially related to the work by Cai et al. (Cai et al., 2011), which proposed methods to evaluate student answer in an intelligent tutoring system. They used LSA and RegEx to assess student answers. Their work showed that the carefully created RegEx had high correlation with human raters' scores. They also noted that the correlation increased when the expected answers created by experts were combined with the previous students' answers to assess new student answers.

3.3 Methods

We developed three different types of classifier models and evaluated their performances separately.

To generate our classifiers, we worked with data from one teacher as she authored an activity in the virtual internship, Land Science. In Land Science, participants work to design a city zoning plan that balances the demands of stakeholders who advocate for indicators of community health. In the activity that this teacher customized, participants describe their proposed zoning changes in a notebook entry. In the first step of our method, the teacher defines assessment

criteria for an entry in terms of core concepts, or the key semantic content they want to be present or absent in an entry. For this entry, the teacher defined five core concepts (see Table 3.1). Next, she constructed six example entries and identified the chunks of text in each example that expressed each concept. In addition, she provided lists of keywords for each core concept that she expected to be present in participant notebook entries.

Afterward, we developed various classifiers for each core concept based on the teacher provided items: sample responses, core concepts, and concept keywords. In this work, we report three such classifier types; The LSA based semantic similarity threshold classifier, the NN based semantic similarity classifier, and the RegEx based classifier.

In both the LSA based and NN based classifiers, we use a sliding window to search for the most similar chunk in an intern's notebook entry. That is, for each teacher-defined chunk, we slide a window of equal size over the student entry. For each such participant-chunk identified by the sliding window over the student's notebook entry, we calculate the semantic similarity of the text within the window to the teacher-defined chunk. After the similarity of all windows to a teacher-chunk has been calculated, we assign the highest value as the similarity score for a given core concept. For LSA based classifiers, we calculated the similarity score using SEMILAR (Rus, Lintean, Banjade, Niraula, & Stefanescu, 2013). For the NN based classifier, we calculated similarity score using the Sent2Vec[1] tool. Since both the tools are capable of taking phrases or sentences as input, we give the chunks as input phrase, hence in the rest of the sections, we call these chunks as phrases.

If the highest similarity score is high enough, e.g. higher than a threshold, we decide the target core concept is present in the student response. Otherwise, we

[1]https://www.microsoft.com/en-us/download/details.aspx?id=52365

infer the student respond does not include the core concept. That is, we developed a semantic similarity based classifier for assessing students' responses.

In order to choose a threshold for the similarity based classifiers, we derived a threshold by calculating the similarity score between the chunks of each of the core concepts tagged by the teacher for both LSA based and NN based methods. See the experiment section for details.

To test the validity of our approach, we developed classifiers for each target concept and then tested them using 199 participant entries coded by humans for the presence or absence of each core concept.

Because our initial thresholds were created without the aid of participant data, we expected that better thresholds would exist. We therefore sought to compare the performance of our classifiers using two different thresholds, the derived thresholds above and ideal thresholds (described in more detail below). To calculate the ideal threshold for each classifier we varied the semantic similarity thresholds from zero to one and obtained precision and recall measures for each threshold using participant data.

For the RegEx based classifiers, we used the teacher provided keywords, which were generated without using participant data, to create regular expression lists for each core concept. We infer that the target core concept is present in a given entry as long as any of its associated keywords are present, as determined by regular expression matching. Therefore, in contrast to the LSA and NN models, a threshold is not required for the RegEx classifiers.

The semantic similarity approach minimizes the teachers' input which encouraged us to adopt it for assessing participant responses with respect to containing (or not) targeted, required concepts. This method is also relatively easy to automate, meaning that after the teacher has made a small set of specifications, classifiers can be developed without further human input. The RegEx approach is

less flexible compared to the semantic similarity approach as novel expressions of a core concept, not encoded yet in the regular expressions, are less likely to be correctly identified. However, the RegEx is capable of identifying core concepts that are characterized by a closed set of keywords and semantic similarity may not be able to perform as needed.

3.4 Experiments and Results

First, we describe the data set we used in our experiments and then present the results obtained with our automatically generated classifiers. We also apply these classifiers to participant generated notebook entries to assess the performance of our models on unseen data.

3.4.1 Dataset

As we mentioned above, our classifiers were generated from specifications made by a teacher as she customized an activity in Land Science. To evaluate our method and test how our classifiers would perform on unseen data, we selected 199 participant entries from prior, uncustomized, implementations of Land Science. We took these entries from uncustomized versions of the activity the teacher in this study worked to customize. In this case, the customizations to this activity's notebook requirements and assessment criteria, as defined by the core concepts, were not drastically different from the requirements and criteria of the original activity. Thus, this situation provided a case where we could test our classifiers on data that was expected to contain some distribution of the core concepts. In general, however, our method for generating classifiers is meant to accommodate both small customizations, such as we have here, and more drastic ones, such as a case where a teacher creates an entirely new activity. Therefore, we cannot always expect to have such similar data for testing.

The 199 participant entries were manually coded for each core concept by two raters. Both raters had worked with the teacher in this study to define the core

concepts and had extensive prior experience coding notebook entries from Land Science. Using the process of social moderation (Herrenkohl & Cornelius, 2013), the raters agreed on the presence or absence of each core concept for each of the 199 entries. From Table 3.1, we see that the distributions of some concepts are balanced (C2), while others are skewed (C5). However, because we built classifiers based on the textual features of teacher samples, skewness should have a small effect on the performance of the model.

Table 3.1: Distribution of concepts in dataset

Concept	Notations	#Concepts	%Concepts
land use changes	C1	141	72.860
original land use configuration	C2	114	57.280
location of land use change	C3	79	39.690
indicator changes	C4	128	64.320
stakeholder demands	C5	46	23.110

3.4.2 Threshold Initialization Method

To derive a similarity score threshold, which is needed for the semantic similarity based classifiers, we calculated the similarity scores between the tagged chunks of text for each core concept in the teacher provided examples. Next, we calculated the average and standard deviation of these scores and set our threshold as the average similarity minus one standard deviation for each core concept. The values we obtained using this approach are reported in Table 3.2, where the last column is the derived threshold for each classifier. Table 3.2 shows thresholds for both LSA based similarity and the NN based model.

Phrase similarity based on LSA relies on the combination of constituent words a phrases. Hence the similarity score will be more biased towards phrases having common words. While the NN based semantic similarity method (P.-S. Huang et al., 2013; Shen et al., 2014)projects the phrase pairs into common low dimensional space hence the similarity score obtained will be more consistent irrespective of the presence of common words in the phrases.

Table 3.2: Derived threshold for LSA based and NN based similarity method

Classifier		Avg.	Std.	Avg. - Std.
C1	LSA	0.584516	0.228474	0.356042
	NN	0.437065	0.122893	0.314172
C2	LSA	0.239488	0.189726	0.049762
	NN	0.242053	0.168682	0.073372
C3	LSA	0.696795	0.103681	0.593114
	NN	0.523347	0.077424	0.445923
C4	LSA	0.278877	0.170271	0.108607
	NN	0.174579	0.124677	0.049902
C5	LSA	0.466482	0.196369	0.270113
	NN	0.149499	0.096005	0.053494

Note: Avg.=average similarity score, Std=standard deviation.

In Table 3.2 it is also observed that the standard deviations of similarity scores for NN based models are less than that of the LSA based semantic similarity model in all the five classifiers. This validates our previous understanding that LSA based similarity measures is more biased towards phrases with high degree of word overlap and gives lower score for the phrases with lower degree of or word overlap, resulting high variation in the score. On the other hand, NN based method does not suffer from such biasedness.

3.4.3 Results

We now present precision and recall results for LSA based and NN based models for the derived thresholds presented earlier and for ideal thresholds (described next). Afterward, we present results for the RegEx based classifiers.

As an alternative to deriving classifiers based on teacher-specified input, we wanted to see how well our methods performed when trained on actual, participant data. That is, when the threshold used in the classifiers to make the final decision was fit based on actual participant data. We call such participant data-trained threshold, the ideal threshold. This ideal threshold could only be computed when participant data is available, which is a major constraint when developing a new internship, as we pointed out earlier.

Figure 3.1 and 3.2 shows the precision and recall plot for increasing

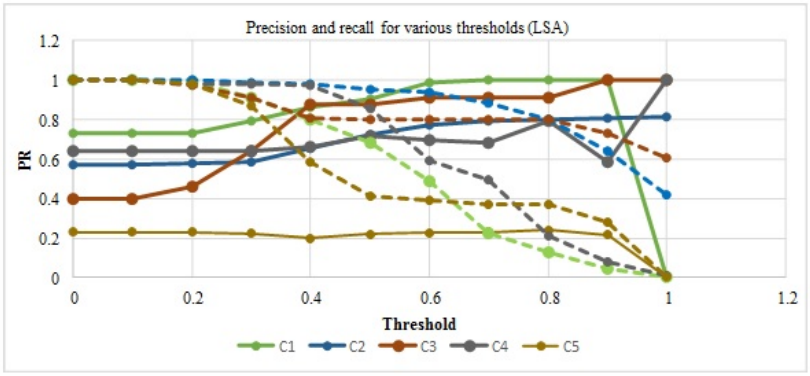

Fig. 3.1: Precision and recall for LSA based similarity thresholds (solid lines are precision; dotted lines are recall)

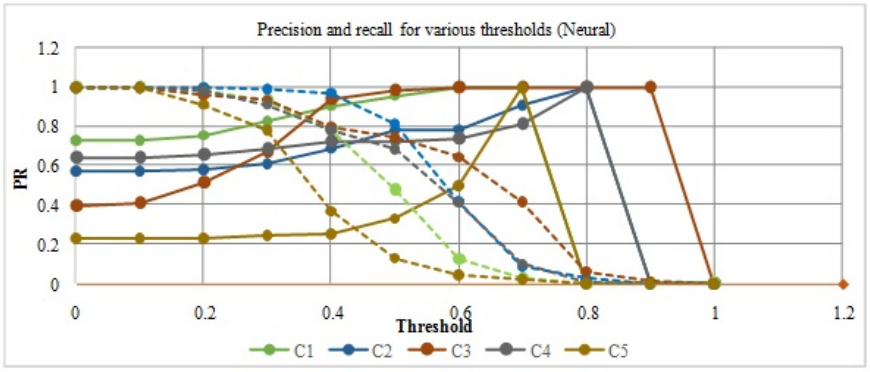

Fig. 3.2: Precision and recall for neural network based similarity thresholds (solid lines are precision; dotted lines are recall)

thresholds of LSA based and NN based similarity methods. These plots were obtained by comparing the model classifications to the manual classifications on the 199 participant entries. It is generally seen that whenever precision increases at a particular threshold, the recall decreases or vice versa. The point of intersection of the precision and recall for a particular classifier gives the ideal precision and recall—that is, the classifier has balanced performance in terms of precision and recall. From the figure, it is clear that if we want fewer false negatives, for example, the value of the threshold should be increased. In such a case, the precision will be compromised. Therefore, the threshold should be chosen carefully not to compromise either precision or recall to an undesirable extent.

The results obtained with ideal and derived thresholds are summarized in Table 3.3. These data suggest that, for the ideal thresholds, the LSA based classifiers for core concepts C1 through C4 performed well with the lowest precision and recall value being 0.72. However, the NN based classifiers outperformed the LSA classifiers for all core concepts other than C2. LSA based models depend on the overlapping content words in phrases and the performance suffers in cases where the phrases contain out of vocabulary words. Out of vocabulary here means the LSA similarity relies on pre-built vocabulary from a large corpus that does not contain some of the words, such as proper nouns that are specific to Land Science. However, NN based similarity models rely on letter trigrams from a very large corpus, and every input phrase is converted to letter trigrams. Therefore, the NN based models are capable of capturing the semantics even when there are out of vocabulary words in the phrases or context of the phrases. Hence, the NN based classifiers are superior for these concepts. However, for C2, the NN based classifier lagged in performance by 2% in precision and recall compared to the LSA based classifier because the teacher samples used for C2 contained only short phrases with very few context words and some of the overlapping words in the phrases boosted LSA based classifiers. The classifier C5 performed poorly for both LSA and NN based classifiers.

For the LSA based classifiers, the highest precision using derived thresholds was 0.92 with recall of 0.80 for C3 and the lowest precision was 0.22 with recall of 0.98 for C5. As we saw with the derived thresholds, NN based classifiers generally outperformed their LSA based classifiers counterparts, with the exception of the recall for concept C3.

The results in Table 3.3 suggest that a good threshold could be derived without participants' data. The high recall and precision using derived thresholds for concepts C1 and C3 suggest the possibility of assessing the core concepts in

Table 3.3: Precision and recall for ideal and derived thresholds for LSA based and NN based similarity method

Classifier		Threshold		Precision		Recall	
		I	D	I	D	I	D
C1	LSA	0.36	0.35	0.84	0.82	0.84	0.86
	NN	0.34	0.31	0.86	0.84	0.86	0.92
C2	LSA	0.80	0.05	0.80	0.57	0.80	1.00
	NN	0.52	0.07	0.78	0.57	0.78	1.00
C3	LSA	0.38	0.59	0.82	0.92	0.82	0.80
	NN	0.36	0.44	0.86	0.96	0.86	0.78
C4	LSA	0.56	0.11	0.72	0.64	0.72	1.00
	NN	0.46	0.05	0.74	0.64	0.74	1.00
C5	LSA	1.00	0.27	0.00	0.22	0.00	0.98
	NN	0.80	0.05	0.00	0.23	0.00	1.00

Note: I=ideal, D=derived.

participant notebook entries with classifiers generated using only the teacher's sample responses. However, when compared to the results using the ideal thresholds, classifiers C2, C4 and C5 did not perform well; their derived thresholds differed largely from their ideal thresholds, and their precision and recall suffered. The relatively low derived threshold values for these concepts suggests that their associated examples, which were used to calculated the thresholds, were semantically dissimilar. Dissimilar examples for a given concept could imply an ill-defined concept and that the provided examples do not represent it well. Alternatively, dissimilar examples could imply a complex or varied concept that requires highly different examples to represent it fully. Because we cannot distinguish between these cases automatically, we plan in future work to set a best guess threshold of 0.5 in such cases.

Table 3.4: Performance of regular expression model

Concepts	Precision	Recall
C1	0.963	0.551
C2	0.640	1.000
C3	1.000	0.746
C4	0.791	0.890
C5	0.894	0.739

Table 3.4 shows the precision and recall of RegEx based classifiers. Here the performance for concepts C2, C4, and C5 is more interesting when we compare those values with the previously discussed result. For example, the precision and recall for C5 improved impressively with values 0.89 and 0.73 respectively, whereas in previous case those values were either undefined or 0 precision with recall 1. Furthermore, the precisions of C1 and C3 are high, however the recalls are relatively low. Qualitatively investigating these results suggested that participants entries expressed these concepts in a variety of ways that were not captured by the regular expression lists.

Given that we see improvements for some core concepts using the regular expression based approach, these results suggest that the teacher provided samples on which the similarity measures where based may not have included a variety of key terms that could indicate the presence or absence of these core concepts. Comparing the sample responses and the keywords provided revealed that the samples indeed did not contain many of the keywords in the list. In some cases, the keywords were synonyms or other instances of particular kinds of words provided in the sample responses. For example, in Land Science, there are sixteen stakeholders who give demands on zoning plans. The core concept C5, stakeholder demands, is meant to capture references to these 16 stakeholders in participant notebook entries. Examining the teacher provided samples, we found that only four stakeholders were covered, while the keyword list for the core concept mentioned all sixteen. We plan in future experiments to either ask teachers to provide enough samples to cover finite sets of semantic content such as this or to incorporate the provided keyword list into the semantic similarity methods as extra samples.

3.5 Conclusions

In this work, we investigated a method for creating classifiers for virtual internship notebook entries using teacher provided specifications without the use of

participant data. Our classifiers used LSA based and NN based semantic similarity methods to capture the general semantic relationships among concepts. We also investigated regular expression based classifiers. The results are impressive in the sense that some classifiers, using both LSA and NN, gave high precision and recall values using thresholds derived without participant data, which suggests that our general method is plausible.

Furthermore, the superiority of the NN classifiers over the LSA classifiers suggests that NN methods are preferable when the participant responses vary widely in terms of style, content, and word overlaps with the teacher provided sample response.

The improved performance for some core concepts, such as C5, using regular expression based classifiers implies that such classifiers performed better for concepts whose sample responses did not contain a variety of keywords, despite the benefits we saw for NN models. These results suggest that, in some cases, teachers may need to provide more exhaustive samples, and that provided keywords and regular expression based classifiers may supplement a semantic similarity approach.

In future work, we will investigate a method to combine the classifiers in order to better understand how performance of one model is boosted by another in the scenario where participants responses vary widely compared to the sample responses. We will also see how the performance be affected by setting up the thresholds to 0.5 for concepts C2, C4 and C5.

Our work has several limitations; most obviously, we used participant data in to evaluate the performance of some of our classifiers. In the real use case of our method, we cannot expect to have such data available. We want to make clear, however, that our purpose in using participant data was not to train better classifiers, but to evaluate our method for generating them. Thus, our results

suggest that this method can produce classifiers that would perform well on unseen data, but more refinements are needed.

Chapter 4

Effect of Corpus Size and LSA Vector Dimension in Assessment

Semantic similarity is a major automated approach to address many tasks such as essay grading, answer assessment, text summarization and information retrieval. Many semantic similarity methods rely on semantic representation such as Latent Semantic Analysis (LSA), an unsupervised method to infer a vectorial semantic representation of words or larger texts such as documents. Two ingredients in obtaining LSA vectorial representations are the corpus of texts from which the vectors are derived and the dimensionality of the resulting space. In this work, we investigate the effect of corpus size and vector dimensionality on assessing student generated content in advanced learning systems, namely, virtual internships. Automating the assessment of student generated content would greatly increase the scalability of virtual internships to millions of learners at reasonable costs. Prior work on automated assessment of notebook entries relied on classifiers trained on participant data. However, when new virtual internships are created for a new domain, for instance, no participant data is available a priori. Here, we report on our effort to develop an LSA-based assessment method without student data. Furthermore, we investigate the optimum corpus size and vector dimensionality for these LSA-based methods.

4.1 Introduction

Semantic similarity is about determining whether two texts (documents, paragraphs, or words) are similar in their meaning. Often the semantic similarity methods represent the documents or terms using a vectorial representation and then apply a similarity function such as computing the cosine of the angle between the corresponding vectors of the documents. The cosine is equivalent to the normalized dot product of the two vectors thus quantifying to what degree the two vectors are

close to each other. The similarity score obtained with such methods depends upon the vectors used in the calculation. The vectors are derived based on a statistical analysis of a large corpus of documents. The analysis produces a term-by-document matrix in which terms represent the rows and documents the columns. Each cell in the matrix indicates, for instance, how frequent the corresponding term in the row is in the corresponding document in the column. Latent Semantic Analysis (LSA) uses Singular Value Decomposition (SVD) to map such high-dimensionality term-by-document matrices onto reduced-rank matrices with the added benefit of being able to capture second order semantic relationships among words (Landauer, Foltz, & Laham, 1998; Bradford, 2008).

Studies have shown that the rank (number of dimensions) of the LSA semantic space and therefore of the LSA vectors (Landauer et al., 1998; Bradford, 2008) as well as the corpus size and its nature (Kontostathis, 2007; Crossley, Dascalu, & McNamara, 2017) influence the quality of the resulting vector representations. Landauer et al. (Landauer et al., 1998) noted that a vector dimension of 300 obtained from moderate corpus sizes performs best in general. However, obtaining the optimum corpus size and vector dimensionality for a particular domain of interest and task is yet to be determined. In this study, we investigate the optimum domain corpus size and vector dimensionality to develop LSA based assessment methods for virtual internships.

4.2 Background

4.2.1 Virtual Internships and Automated Assessment

During virtual internships, students work on various tasks, e.g., engineering design tasks. While working on these tasks, they need to provide justifications for their work, e.g., justifications for their designs, in digital notebooks. The notebook entries are then assessed as acceptable or unacceptable by human raters. Automating this assessment task is an important step towards reducing the time

and cost associated with developing and running such virtual internships. Previous attempts to automatically assess notebook entries in virtual internships focused on automated classifiers trained on participant data collected after the initial development and deployment of the corresponding virtual internship. This means that one needs to run the virtual internship at the beginning using human graders, which is tedious, time-consuming, not-scalable, and expensive. The major challenge to developing automated assessment methods from the very start is the fact that participant data is not available at the time when a new internship is being developed, e.g., for a completely new set of tasks or a completely new domain (Swiecki & Shaffer, 2017)). For instance, when instructors create new activities or customize existing activities for an existing internship, previously trained assessment classifiers become invalid. Indeed, customization makes trained clasifiers invalid whereas new internships for new domains leaves the virtual internship system without classifiers until learner data is collected. Our work is motivated by this need to develop automated assessment method early on, at design time, for a new virtual internship when participant data is not yet available. To this end, we explore methods to generate classifiers without student data. Such approaches would enable the development of virtual internships that incorporate automated assessment methods from the very beginning, avoiding the need to have human raters assess learner responses during initial deployment of such new virtual internships. We further extend prior work (Gautam, Zachari Swiecki, Graesser, & Rus, 2017) by exploring optimum corpus size and the LSA vector dimensionality for student notebook assessment.

4.2.2 LSA and its Use in Text Analysis

LSA is a vector space model for deriving semantic representations of words and largers texts such as documents. It relies on a reduced-rank approximation of a term-document matrix which captures word co-occurrence information from natural

texts such as textbooks. The rank reduction is used to remove the noise introduced by sparsity of the term document matrix. Jessup and Martin (Jessup & Martin, 2001) showed that the optimal rank choice delivers improvement in performance in an information retrieval task. Using LSA vectors instead of using standard term frequency - inverted document frequency (tf-idf) vectors in a retrieval system, the system will be able to retrieve documents based on concepts as opposed to keywords, which improves the recall of the system (Bellegarda, 2005). It should be noted that, usually, when recall increases precision decreases.

In addition to the rank (vector dimension), the size of the input corpus also has an impact on the performance of LSA vector spaces. A study conducted by Crossley et al. (Crossley et al., 2017) showed that LSA space developed using larger corpora performed better than when smaller corpora was used in a word association task and a vocabulary level test. It should be noted that they developed the LSA space using a multiple domain corpus (TASA[1]) and a single domain corpus from the Corpus of Contemporary American English (COCA) (Davies, 2010). Furthermore, Landauer et al. (Landauer et al., 1998) generated an LSA space from encyclopedia articles and used it for a Test of English as a Foreigh Language (TOEFL[2])word comparison task. They found that the vectors with dimensionaliy of 300 performed best, which was considered a standard number of dimensions for many applications.

LSA has been used successfully in various tasks such as answer grading, text summarization, e-mail categorization, and information retrieval. For instance, LSA based essay grading yields comparable performance to human graders. Landauer and colleagues compared the human ratings of passages written by students with LSA-obtained rating and found that the meaning of passages could be carried by the words independent of their order, which is what LSA is about (Landauer, Laham, Rehder, & Schreiner, 1997). In another study, (Mohler & Mihalcea, 2009)

[1]http://lsa.colorado.edu/spaces.html
[2]www.ets.org/toefl

showed that an LSA based short answer grading method performed as well as a knowledge based method, e.g., methods that rely on knowledge-based resources such as WordNet. The LSA based approach has a major advantage over the knowledge based approach - the LSA model could be constructed automatically, in an unsupervised manner, whereas knowledge based approaches, e.g., such as those relying on WordNet, require extensive manual efforts to build the knowledge based resources.

Pérez et al (Pérez, Alfonseca, et al., 2005; Pérez, Gliozzo, et al., 2005) obtained LSA space from a large collection of pre-categorized domain corpus and combined with the BLEU algorithm, a method used to evaluate machine translation (Papineni, Roukos, Ward, & Zhu, 2002), to assess student's freely generated textual answers. They claimed that their method achieved state-of-the-art correlations to teachers' scores. In their method, they averaged word vectors to represent student and reference answers. It should be noted that, though our vector representation approach is similar, we have collected domain corpus from Wikipedia by automatically filtering the Wikipedia articles.

Other uses of LSA relevant to our work are from text summarization. LSA based methods have been shown to extract better summaries (Gong & Liu, 2001; Steinberger & Ježek, 2004; Yeh, Ke, Yang, & Meng, 2005), particularly, when choosing appropriate vector dimensions, LSA helps to extract better semantically similar summaries from original documents when compared to keyword based summaries (Yeh et al., 2005).

Another related method was proposed by Dredze and colleagues (Dredze, Wallach, Puller, & Pereira, 2008) to generate keywords for e-mail messages without annotated training data. E-mail summary keywords are generated to represent e-mails in e-mail filtering systems. The keywords act as features for e-mail classification. In the Dredze and colleagues' method, they generated LSA vectors to

represent each word in e-mails and identified each word as a keyword if the word was present in an e-mail. They claimed that the LSA based method provided a good representation of e-mails compared to tf-idf based approaches.

Like these approaches above, our method uses LSA for assessing learners' free responses in learning environments such as virtual internships. We explore the impact of the corpus size and space dimensionality on the performance of the resulting assessment method. This work leverages the previous approach (Cai et al., 2018) of extracting domain corpus. However, it should be noted that the the previous work analyzed the learners' responses that were few words to a sentence length. Whereas in this work we further extend previous study by analyzing the learners' response consisting of a paragraph of two to four sentences.

4.3 Our Method

Our method involves a two step process. First, we develop core concept identifiers based on a small set of seed data provided by teachers when developing a new internship or new tasks for an exiting internship. Second, we evaluated the performance of these classifiers as a function of the corpus size and dimensionality of the semantic space. We describe next the method and the corpus we used in our experiments.

4.3.1 Classifier Generation

In the first step of our method for generating classifiers without student data, teachers define a small set of seed concepts for a given task, e.g., an engineering design task. The seed concepts represent the key semantic content student notebook entries should contain. In addition, teachers provide a small set of exemplar or benchmark notebook entries in which they tag the seed concepts.

Using this seed information we develop a semantic similarity based identifier for each seed or core concept. That is, we try to identify whether a target core concept is present in the student answer or not. There is one identifier for each of

Table 4.1: Algorithm to obtain average similarity score between the chunks of a concept in exemplars

Input: *Set C of annotated chunks for a concept in exemplars*
Output: *Average A and standard deviation SD of similarities between chunks of a concepts in exemplars*
Initialize: S = empty list of similarities
Initialize: P = empty list of set of chunk pairs
do for each c_i *in C:*
  do for each c_j *in C:*
    $p = set(c_i, c_j)$
    if $c_i \neq c_j$ *and p not in P:*
      $S = S + similarity(c_i, c_j)$
      $P = P + p$
A = Average(S)
SD = Standard deviation(S)

the core concepts. The identifiers are modeled as classifiers. Each of the classifiers uses a sliding window to search for the chunk of text in a participant notebook entry that is most similar to the target core concept. For this purpose, we use a sliding window of size equal to the length of the core concept. The window slides over the participant response and for each instance, we calculate the similarity between the chunk of text in the window and a target core concept. We select the chunk of text in the student notebook entry that corresponds to the highest similarity score.

If the highest score is higher than a threshold, we decide that the target core concept is present in the student response. In order to obtain the threshold for a target core concept, we calculated similarity scores between all possible tagged chunk pairs for that concept in the teacher generated ideal responses, ie., exemplars. Then, we computed the threshold as one standard deviation below the average similarity score. The detailed steps for obtaining the average and standard deviation are provided in Table 4.1.

To calculate the similarity score between core concepts and the sliding window, we use an LSA-based semantic similarity implementation available in

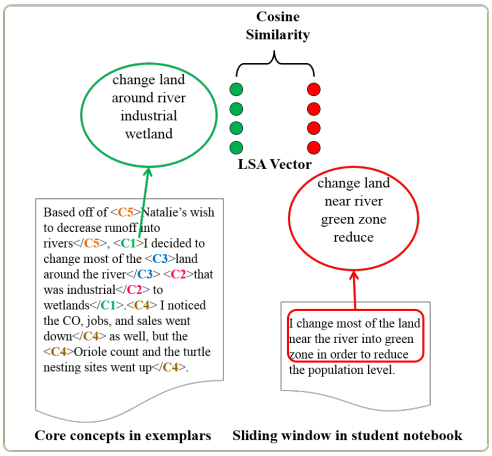

Fig. 4.1: Concepts in exemplars and sliding windows

SEMILAR (Rus, Lintean, et al., 2013). Figure 4.1 shows the overview of obtaining cosine similarity scores between a core concept and the chunk of texts corresponding to one instance of the sliding window.

In our previous work (Rus et al., 2016), we obtained word vectors from the TASA corpus. The corpus consists of contents from textbooks, literature works. This domain general LSA model was developed from about 38,000 documents and 92,000 terms (Stefănescu, Banjade, & Rus, 2014). We compared the performance of our classifiers using the TASA corpus to the performance obtained with a domain specific corpus, which is described next.

4.3.2 Domain Corpus Collection

To collect the domain specific corpus, we started with a seed corpus of a small number of documents from our target internship, *Land Science*. These documents include text resources about urban planning and instructions sent to participants during the virtual internship. Next, we extracted keywords from this seed corpus and assigned a "keyness" value to each keyword. The keyness depends upon two factors: first, if a word occurs frequently in domain general corpus (such as TASA), then the word is less important for a particular domain. Second, if a word occurs frequently in a domain specific corpus, the word is important for that

Table 4.2: Annotation of exemplar for core concepts

Based off of <C5>Natalie's wish to decrease runoff into rivers</C5>, <C1>I decided to change most of the <C3>land around the river</C3><C2>that was industrial</C2>to wetlands</C1>. <C4>I noticed the CO, jobs, and sales went down</C4>as well, but the <C4>Oriole count and the turtle nesting sites went up</C4>.

domain. Hence the keyness value is obtained by taking into account both factors as described in (Cai et al., 2018). An average "keyness" value is obtained based on the keyness values for each word that appears in a Wikipedia document. This average keyness is viewed as the document keyness by which the documents are ranked in order to select the domain specific documents.

While LSA vectors are essential components in our method, the dimensionality of the LSA space and therefore of the LSA vectors as well as the size of the domain specific corpus are important parameters that affect the predictive power as well as scalability of the classifiers we develop. Therefore, in this work we analyze the impact of these parameters on the performance of the assessment classifiers by exploring different corpus sizes and vector dimensionalities and observing their impact on the performance of our classifier in terms of F-1 scores, as explained next.

4.4 Experiments and Results

4.4.1 Dataset

As mentioned, our goal is to study how corpus size, corpus domain specificity, and dimensionality of LSA vectors affect the performance of student answer assessment methods. Our experimental data consists of student responses in virtual internships which were used to evaluate our classifiers. The classifiers classify each sentence from student responses as the presence or absence of a core concept in it.

Table 4.3: Number of core concepts in exemplars and student notebooks

Concepts	Notation	#_E	#_S
Runoff	C1	2	26
Phosphorous	C2	2	15
Orioles	C3	3	25
Housing	C4	4	33
Turtles Nesting Sites	C5	3	17
Jobs	C6	5	45
Sales	C7	2	27
CO	C8	4	17
Justification	C9	11	101
Land use change	C10	13	120
Indicator change	C11	14	92
Indicator value	C12	9	5
Directly quotes information from resource readings or teammate	C13	5	0

Note: #_E for exemplars and #_S for student notebooks.

i Domain corpus: It consists of documents selectively chosen from Wikipedia articles. The corpus consists of 32,000 documents that are relevant to the Land Science virtual internship. From the collection, we randomly selected 2,000, 4,000, 8,000, 16,000 and 32,000 documents to generate 1,000 dimension LSA word vectors. Since the vector dimensions are ranked, we generated 1,000 dimension vectors, instead of generating separate LSA vectors of different dimensions with varying number of documents. Selecting fewer dimensions from 1,000 dimension vector simplifies the LSA vector generation process with negligible information loss.

ii Notebooks: Two sets of of annotated notebooks from Land Science, one with 14 exemplars created by teachers and 100 randomly selected participant notebooks entries. The exemplar notebooks were annotated for the core concepts (see Table 4.2). The 100 notebook entries were split into 550 sentences. These sentences are manually filtered to remove noisy sentences that only consist of

non-english words or out of domain words. After filtering, we were left with 278 sentences, each of was then annotated with the presence or absence of the 13 core concepts. It should be noted that a notebook may consist of a small subset of core concepts, which means some concepts are more likely to appear than the others as seen in Table 4.3. The distribution of concepts in exemplars and student notebooks is seen in Figure 4.2. From the figure, it can be noted that some of the concepts (e.g., C12 and C13) are much more rare in student answers compared to the exemplars. For performance calculation, we do not include concepts (such as C13) which are absent in student notebook.

Besides the domain specific corpus, we also used the TASA corpus in order to compare the performance with the domain corpus.

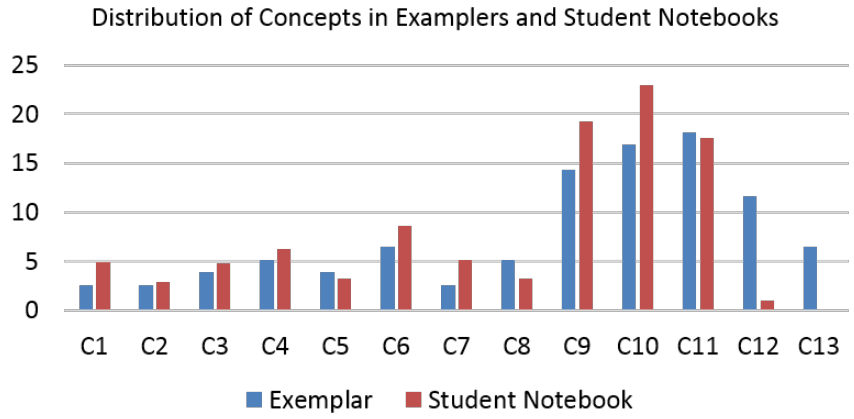

Fig. 4.2: Distribution of core concepts

4.4.2 Results

Figure 4.3 shows the surface plot of average F-1 scores for all core concepts for different combinations of domain corpus size (Spaces) and number of space dimensions (Dimensions). The plot suggests that the performance initially improves as the corpus size increases, up to a certain point, after which the performance starts decreasing. Furthermore, the figure indicates that the performance initially improves with an increasing number of dimensions, stays constant for a little while,

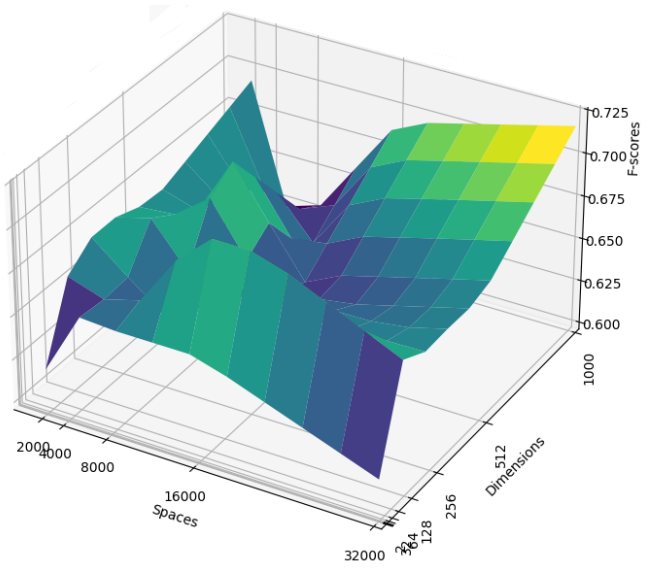

Fig. 4.3: Surface plot of average F-1 scores for domain corpus of different sizes (spaces) and number of dimensions

then starts decreasing and finally increases again. It could be seen that the overall performance is determined by a combination of corpus size and vector dimensionality.

Table 4.4: Minimum and maximum average performance among core concepts for the combination of corpus size and vector dimension

	Metrics	Values	(Space, Dimension)
Minimum	P	0.798	(4000, 1000)
	R	0.600	(32000, 2)
	F	0.620	(32000, 2)
Maximum	P	0.925	(32000, 1000)
	R	0.701	(32000, 1000)
	F	0.716	(32000, 1000)

In the Table 4.4, the minimum and maximum of Precision, Recall, and F-1 scores along with the corresponding combination of corpus size and vector dimensionality for the domain corpus are shown. These scores are obtained by averaging the corresponding metrics for all 13 concepts. From the table, we see that 1,000 dimensions with 32,000 documents performed best(F-1=0.71). It should be noted, however, that even though the performance is better compared to other combinations, other combinations with smaller corpus sizes and fewer dimensions

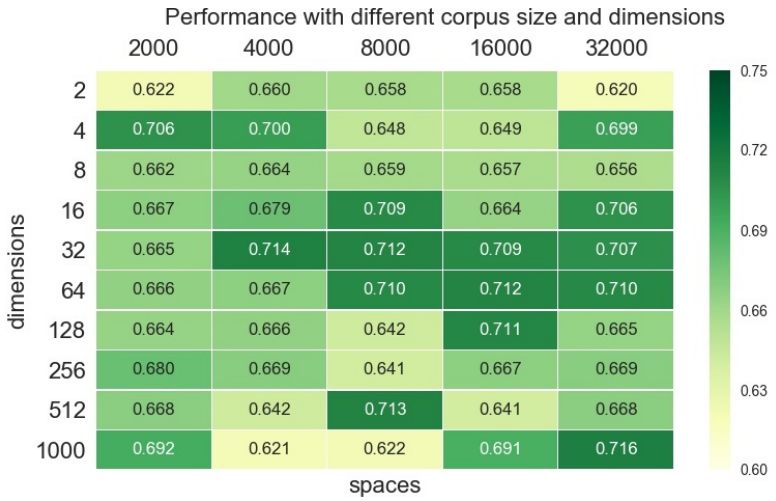

Fig. 4.4: Heatmap for F1 scores with different combinations of corpus size and vector dimensions

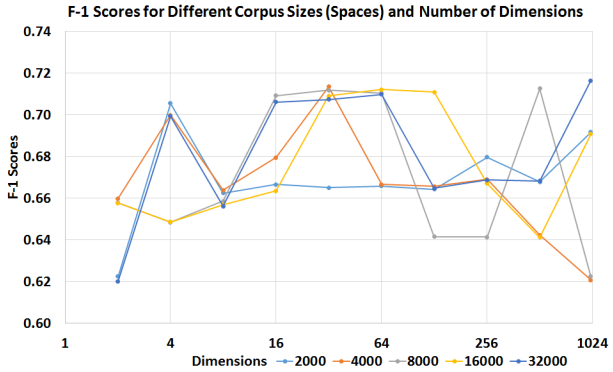

Fig. 4.5: Average F-1 scores for domain corpus of different sizes (see legends in the figure) and number of dimensions

perform comparably well (see Figure 4.5). Figure 4.5 further suggests that the F-1 score initially improves (except for a corpus size of 16,000) when vector dimensionality increases, followed by a drop, and then improves again at a quick rate followed by a plateau in which the performance remains constant. Moreover, it is seen from the heatmap (Figure 4.4) that the performance remains comparatively same with a corpus size of 4,000 through 32,000 documents for dimensionality of 32. The overall performance is almost comparable to the maximum performance (F-1=0.716), suggesting that a small corpus (size=400) and small dimensionality

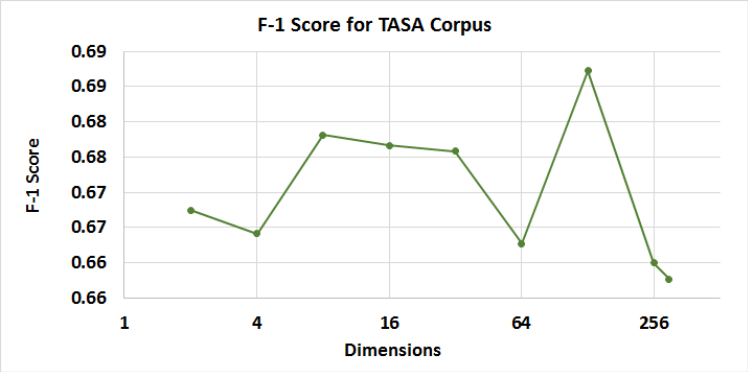

Fig. 4.6: F-1 scores for TASA corpus with varying dimensions

(=32) could be good enough to develop an assessment component for virtual internships.

Figure 4.6 shows the trend of F-1 scores for varying vector dimensions of the TASA corpus. The graph resembles the similar pattern observed for the domain specific corpus, where the performance improves initially, then remains comparatively the same and then starts dropping again. It can be concluded that a small domain corpus can give better performance than the much larger TASA corpus when a proper combination of corpus size and vector dimensionality is chosen.

4.5 Conclusions

We presented in this paper an approach to develop student answer assessment methods without student data. We also analyzed the impact of corpus size, corpus specificity, and semantic space dimensionality on the performance of the assessment methods. Our analysis showed that the LSA spaces generated from a domain specific corpus can perform better when compared to a space generated from the much larger TASA corpus for the notebook assessment task. For the domain specific corpus, the best average performance over all the target concepts was obtained for the maximum available corpus size and the maximum number of dimensions. However, this performance is comparable to results obtained with a smaller corpus size and smaller vector dimensionality indicating that smaller

corpora and spaces can be good enough to boost assessment components for virtual internships. We plan to further experiment with our method on other virtual internships and data from other adaptive learning technologies to see if our current conclusions hold on such new data.

Chapter 5

LSTM Based Negation Handling

Negation plays a significant role in spoken and written natural languages. Negation is used in language to deny something or to reverse the polarity or the sense of a statement. This paper presents a novel approach to automatically handling negation in tutorial dialogues using deep learning methods. In particular, we explored various Long Short Term Memory (LSTM) models to automatically detect negation focus, scope and cue in tutorial dialogues collected from experiments with actual students interacting with the state-of-the-art intelligent tutoring system, DeepTutor. The results obtained are promising.

5.1 Introduction

All human languages have negation as a key linguistic feature. Negation reverses the polarity of entire statements or of parts of statements. (Givón, 1993) categorized negation broadly into two classes: morphological and syntactic negation. The morphological negation follows a specific structure where a root word is modified by prefixes such as "non", "un" or suffixes such as "less". On the other hand, in syntactic negations, explicit negation cues are used to affect the meaning of a single word or a group of words.

Negations are marked by a set of cue words (or negation words) such as "no", "none", "not", "neither", "nor" and their variants(such as " 'nt"). Statements might be negated either explicitly with explicit cues, e.g. "no" or "not", or implicitly such as "avoid", "prevent", "prohibit" and so on. In the latter case, we have implicit negation. The *negation cue* can affect one or more parts of the statement in which it occurs - the affected parts are called the *scope* of negation. The constituent in the scope that is prominently negated is called the *focus* of the negation (Huddleston, Pullum, et al., 2002).

An example of negation is shown below. The negation clue is delimited by <<>>, within square brackets ([]) we show the scope of negation, and within curly brackets {} we show the negation focus.

[There are] <<no>>[{forces} acting upon the puck] because it is at rest.

Many studies showed that negation accounts for a significantly large part of both spoken and written human language. In one study, it is reported that negation occurs twice as often in speech as in writing (Tottie, 1993). Some domain-specific corpus linguistics studies showed that negation occurs most frequently and represents a major portion of the information within such domain specific texts. For example, in a corpus of medical textual documents and biological scientific papers developed by Vincze and colleagues (Vincze et al., 2008), 10% of the sentences contain negation. A customer review corpus of movies, books and consumer products (Konstantinova et al., 2012) includes 18.1% negated sentences. Also, an analysis of student utterances in dialogues collected from experiments with actual students interacting with the intelligent tutoring system DeepTutor (Rus, D'Mello, et al., 2013) showed that 9.36% of student utterances included explicit negation.

In this work, we focus on handling negation in dialogues. Linguistics phenomena, such as ellipsis and pragmatics that are more prevalent in dialogues, make negation more complex to automatically analyze in this case as the meaning of a sentence relies on previous dialogue turns, i.e. previous context. The example below shows five possible answers with explicit (*A1-A3*) or implicit (*A4 and A5*) negation for the same tutor question (*Q*). It should be noted that in this work, we limit our analysis to explicit negation.

Q: *Do a ping-pong ball and a bowling ball falling from the same height hit the ground with same force of impact?*

A1: *No*

A2: *They do not hit the ground with the same force of impact.*

A3: *The respective impacts of the two balls is never the same.*

A4. *The impact of ping-pong ball will be smaller.*

A5. *The impact of bowling ball will be greater.*

Previous automated approaches to negation handling rely on the key observation that both written and spoken texts can be regarded as sequences of events, i.e. words, in which a word occurrence depends upon the long sequence of the previously occurred words. An analysis of such sequences of observations, e.g. to detect the negation scope, could be approached as a sequence labeling task in which each word is labeled as being part (or not) of the negation scope associated with a negation cue word. For instance, probabilistic approaches such as Hidden Markov Models (HMM) or Conditional Random Fields (CRF) have been used for negation handling (Pröllochs, Feuerriegel, & Neumann, 2016; Banjade, Niraula, & Rus, 2016). In these approaches such as HMM, the model becomes complex when long term dependency is taken into account. And in the case of CRF it also requires a set of hand crafted features (in addition to labeled data). Because of recent successes in training deep neural networks, and in particular recurrent neural networks (RNN) which are relevant to our task, and because of their ability to handle sequential data, it makes sense to approach the task of negation handling using a recurrent neural network approach. In this work, we will explore one version of RNNs that account for long term dependencies, namely, Long Short Term Memory (LSTM) RNNs (Hochreiter & Schmidhuber, 1997). Neural networks can perform equally well or sometimes better than other complex sequence labeling models with comparatively large number of parameters or human constructed features. The only human involvement in a neural network based model is the labeling of data.

Specifically, we propose a deep learning approach to identify *negation scope*, *negation focus* and *negation cues* in tutorial dialogues. We explore various LSTM network architectures and compare their performance on a negation handling task

using actual student utterance from several DeepTutor experiments. Furthermore, we explore a better input-output strategy to train and test the model in order to performance, i.e. better handle negation of unseen sequences of words which are future dialogue utterances in our case. While the negation scope and focus detection task is similar to prior work (Banjade, Niraula, & Rus, 2016), it should be noted that the primary motivation and novelty of this work is to explore deep learning methods. Our model detects negation cues as well, which is an additional contribution of our work compared to Banjade and colleagues.

In the next sections, we discuss related works, various models, experiments and results. The paper ends with conclusion and future work.

5.2 Related Works

Negation in natural language has been studied since ancient time (Wedin, 1990). It is worth mentioning Horn's work (Horn, 1989) on the effect and influence of negation in languages. Horn described the construct, usage and cognitive processing of negation. It should be noted that in computational linguistics, negation handling methods were initially studied in the context of medical documents. Mutalik and colleagues (Mutalik, Deshpande, & Nadkarni, 2001) developed a tool called Negfinder to detect negated concepts in dictated medical documents. They hypothesized that the negated concepts could be detected using a lexer and parsers that are generally used to analyze programming languages. In another work, Rokach and colleagues (Rokach, Romano, & Maimon, 2008) proposed a pattern learning method for the automatic identification of the negative context in clinical narrative reports. They proposed several steps including corpus preparation, regular expression pattern learning, and training classifiers to predict negation.

Councill and colleagues (Councill, McDonald, & Velikovich, 2010) developed a Conditional Random Fields (CRFs) based state-of-the-art sentiment analysis system using features from an English dependency parser. In this work, they limit

their study to explicit negation within a single sentence. They also developed a negation corpus of English product reviews obtained from the open web.

Banjade and colleagues (Banjade, Niraula, & Rus, 2016) proposed a sequence labeling model using CRFs to detect the negation scope and focus in dialogues. They also collected and manually annotated about 1,000 dialogues, i.e. student and tutoring system interactions collected from actual students using DeepTutor (Rus, D'Mello, et al., 2013). Their work is the first to use CRFs to automatically detect the negation scope and focus both within the curent utterance and previous utterance. That is, in their case the negation scope could include parts of the previous utterrance (previous dialogue context) besides parts or the whole current student utterance. Though we performed our experiment on DeepTutor dialogues, our approach is different from theirs in two aspects: firstly, we proposed a deep learning method using LSTM and, secondly, the LSTM based models we proposed do not rely on labor intensive feature engineering.

LSTM-based approaches have been succesfully used in various natural language processing tasks. Cho et al. (Cho et al., 2014) proposed a recurrent neural network (RNN) model to encode sequence of input words (a phrase) into a fixed dimension vector. A varient of Cho's model was explored by Sutskever and colleagues (Sutskever, Vinyals, & Le, 2014). They proposed a deep LSTM approach for sequence to sequence learning for machine translation. In their method, a multilayer LSTM (encoder) mapped a sequence of inputs onto a fixed dimension vector and another multilayer LSTM (decoder) constructed the target sequence from the vector. They demonstrated that even with limited vocabulary, their model outperformed statistical machine translation models that use a large vocabulary to translate from English to French. They also found that reversing the input data improved the performance of the model. Since we model our negation handling task

as sequence labeling problem, we explore an encoder-decoder model similar to Sutskever and colleagues.

Recurrent neural networks have been used in predicting sequences such as generating words by using models that predict next letters (Sutskever, Martens, & Hinton, 2011). Motivated by the success of recurrent neural network in sequence generation, Bi-directional LSTM (B-LSTM), a variant of LSTM, have been used in sequence tagging tasks (Z. Huang, Xu, & Yu, 2015). Huang proposed various LSTM based models for sequence tagging. Their experiments on NLP benchmark sequence tagging tasks and using standardized data sets showed that hybrid models that combines B-LSTM with CRF produce state of the art accuracy on parts-of-speech tagging (POS), chunking, and named entity recognition (NER).

In many applications that need automated natural language understanding such as intelligent tutoring systems or question answering systems, challenges arise when dealing with negation such as accounting for context. Widdows and Peters (Widdows & Peters, 2003) proposed a model of vector negation in which the portion of the negation vector was subtracted from the vector representation of the document, which captures in some ways the full context of the document. RNNs, on which our approach relies, account for previous context using memory elements.

A critical ingredient in machine learning approaches to negation handling is collecting relevant data. There were several attempts to develop negation corpora in different domains. Vincze et al. (Vincze et al., 2008) developed the BioScope corpus which consists of biomedical texts. BioScope contains annotations at token level for negative and speculative keywords and their scope within a sentence. Konstantinova et al. (Konstantinova et al., 2012) developed an annotated corpus for negation and speculation in review texts. The corpus consisted of 400 documents of movie, book and consumer product reviews. They annotated negative and speculative keywords at token level and their scope (at sentence level). Morante and colleagues (Morante,

Schrauwen, & Daelemans, 2011) published a comprehensive guideline for annotating negation cues and their scope. Blanco and Moldovan (Blanco & Moldovan, 2011) proposed a method to semantically represent negation using focus detection. They extracted and annotated negation focus on texts extracted from PropBank. The annotated data set was used in one of the shared tasks held in 2012 (*SEM 2012) which was about resolving the scope and focus of negation (Morante & Blanco, 2012). These data sets do not fit our goal of handling negation in dialogue. Instead, we will use the data set prepared by Banjade and colleagues (Banjade, Niraula, & Rus, 2016) which contains annotated tutorial dialogues.

5.3 Model Description

We model the negation handling task as a sequence labeling task where a sequence of input tokens, i.e. words, need to be labeled within *scope/focus* and *cue* tags. Unlike previously proposed sequence labeling methods such as HMM or CRF, our LSTM-based approach is better suited for identifying the scope and focus of the negation cue in dialogues because the scope and focus could extend to previous dialogues utterances relative to the utterance where the negation cue is. In such cases, a model able to take into account long term dependencies should be preferred; that is, a long term memory based model such as LSTM.

More specifically, in our approach a sequence of tokens provided as input is processed and a sequence of tags, one for each input token, is produced as output. During training, input sequences are constructed from labeled conversation data by sliding a specific sized window over each utterance. Since the LSTM requires training sequences to be of equal length and the average size of our corpus is roughly 10 tokens per utterance, we set the window size to be of 5 tokens. In order to preserve the continuity of the context tokens that are in the original dialogue, we slide the window one step at a time so that each consecutive window overlaps to the previous window. Also, we slide the window over the labels exactly in the same way

to obtain a sequence of labels for the corresponding tokens in the input utterance so that each input sequence is paired with the corresponding label sequence. For sequences shorter than the window size, a special padding symbol is appended both to the input and label sequences.

Figure 5.1 & 5.2 show our two different LSTM network architectures for negation handling. In the figures, the LSTM blocks from left to right show the unrolled network over time (t_1 through t_n). The output sequence is seen as the probability (given by softmax layer) of each label once the labels for all the tokens are processed (confirmed by time distributed dense layer). The labels (tags) corresponding to the input tokens are predicted by decoding the probability distribution given by the softmax layer.

In the following subsections, we describe in more detail the two different model architectures namely Sequence to Sequence Tagger and Tag Sequence Generator we designed for negation handling.

5.3.1 Sequence to Sequence Tagger

Similar to various sequence to sequence architectures proposed previously, our sequence to sequence architecture uses two different layers of LSTMs, one for encoding the input sequence into an embedding vector and the other to decode the embedding representation onto an output sequence. In general, such encoder-decoder architecture could be used to map an input sequence to an output sequence whose length may not be equal to the input. However, our task is to label each word with a negation tag, hence the output tag sequence is same length as input sequence. Instead of feeding the previous output back to its input, as in (Sutskever et al., 2014; Cho et al., 2014), we feed copies of the vector from the encoder to each of the LSTM units of decoders. Since each LSTM unit uses its state from the previous time step, simply feeding a copy of encoder output simplifies our model without loss of generality of the model. The Repeat layer (Figure 5.1) does

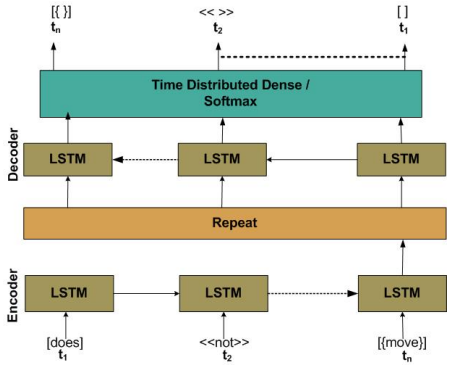

Fig. 5.1: LSTM network of encoder-decoder model for predicting labels of tokens sequence

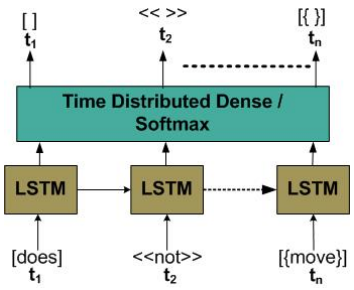

Fig. 5.2: LSTM network of label sequence generator

the job of copying the final output of encoder layer to the decoder layer at every time step of decoding.

5.3.2 Tag Sequence Generator

Recurrent neural networks were successfully used in language models to generate next letter or word based on long term context (Sutskever et al., 2011; Graves, 2013). Unlike sequence to sequence architecture, where the input sequence is mapped to output sequence of arbitrary length, this architecture generates one tag for each input word. Since this architecture does not have a decoding LSTM layer, it generates a label immediately after receiving an input token. Similar to previous architecture, a time distributed dense layer is used (Figure 5.2) to ensure that a sequence of labels is obtained once all the tokens in the input sequence have been seen.

5.4 Experiments and Results

We performed two categories of experiments with the above architectures for various model configurations presented in Tables 5.1. In the first category, we used one-hot-encoding vector representation of each token in the dialogue utterances. In the second category, we used word2vec (Mikolov, Chen, Corrado, & Dean, 2013), a distributed word vector representation (word embedding) for each token in the dialogues. In order to represent words that are not present in the word2vec model, we used a vector average of the synonyms of the corresponding word. Also, if synonyms are not available, we randomly initialized the vector for that word. The performance of the models were evaluated using standard performance measures such as precision, recall and F-1 scores obtained using a 10-folds cross-validation methodology. Results were obtained for identifying the *cue*, *scope* and *focus*, respectively. While annotating our tutorial dialogues, we followed annotation guideline proposed by Morante and colleagues (Morante et al., 2011; Morante & Blanco, 2012), however, in constrast to their data, which was extracted from PropBank and consists of non-dialogue texts, our dataset contains dialogue data.

5.4.1 Dataset

As mentioned, we use an annotated corpus extracted from the set of dialogues between a computer tutor (DeepTutor[1]) and high-school students. During the tutor-tutee interactions, students are challenged to solve conceptual physics problems and if they struggle then a scaffolding dialogue is initiated in which the computer tutor is trying to help the student solve the problem based on socio-constructivist theories of learning. As previously noted, 9.36% of the student utterances in the dataset contain at least one explicit negation cue word such as *no* and *not*.

[1] www://deeptutor.org/

Table 5.1: Configurations of models

Model	Configuration
M1, M2	• Single layer of both encoder and decoder with 150 LSTM units in each • No dropouts • M1 has training batch size = 100, M2 has batch size of 50
M3, M4	• **M3**:Single layer of both encoder and decoder with 150 LSTM units in each • **M4**:Two layers with 150 LSTM units in each layer of both encoder and decoder • Dropouts = 0.3 for each unit • Training batch size = 50
M5	• Two layers with 150 LSTM units in each layer • Dropouts = 0.3 for each units • Training batch size = 50

Note:M1, M2, M3 and M4 are Sequence to Sequence (encoder-decoder) model and M5 is Sequence Generator model

The DT-Neg corpus consists of 1,088 instances of student utterances with at least one negation cue word. Each of these utterances were manually annotated with negation cue word, negation scope and negation focus. As shown in the examples below, the scope and focus could be in same dialogue utterance (Example 1) or in the previous dialogue utterance (Example 2).

Example 1: Q: According to Newton's first law, if the parachutist moves with constant velocity what is the net force acting on the parachutist?

A: <<no>>[{net forces}]

Example 2: Q: According to Newton's first law, if the parachutist moves with constant velocity [what is the {net force} acting on the parachutist]?

A: <<no>>

About 42% of the instances in the DT-Neg corpus have the scope and focus located in the previous dialogue utterance (i.e., dialogue context).

5.4.2 Results

Tables 5.2, 5.3 and 5.4 show the average precision, recall and F-1 scores based on 10-folds cross validation for five different model configurations. In average, M4 has the highest F-1 scores for the detection of *focus*(.839), *scope*(.857) and *cue*(.995) when one-hot-encoding is used for input sequences. Also M4 exhibited best F-1 scores when word embeddings for input sequences were used. From Table 5.3, 5.2 & 5.4, it can be seen that the F-1 measure impoves from first model to the fourth when one-hot-encoding was used. In fact, we experimented with more configurations, however we present only the best ones here. The improvement in the F-1 score for scope and focus detection for M3 shows the positive effect of using dropout on the generalization power of the underling model. Adding one more layer for encoder-decoder slightly improved performance. For focus detection, the F-1 score (0.839) is higher than that obtained (0.826) in previous work by Banjade and colleagues (Banjade, Niraula, & Rus, 2016).

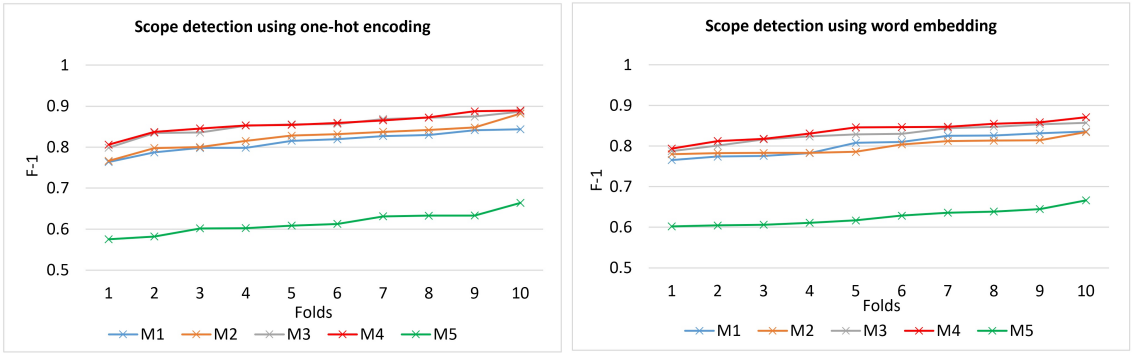

Fig. 5.3: F-1 scores of 10-folds cross validations of different models for scope detection

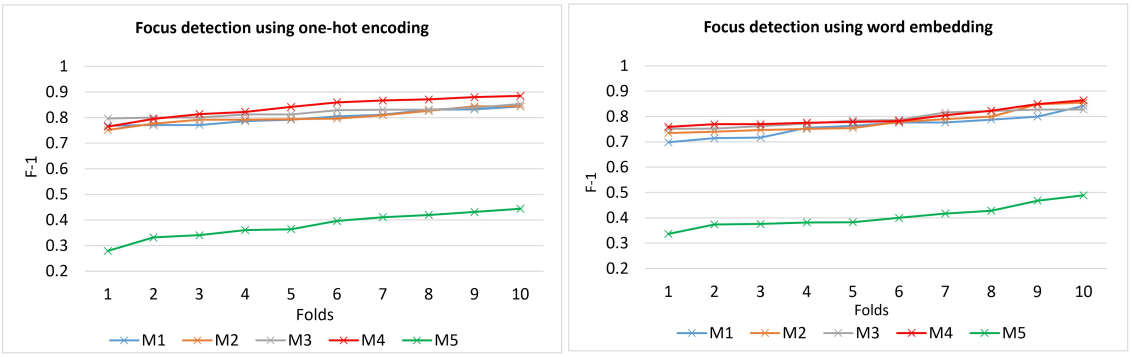

Fig. 5.4: F-1 scores of 10-folds cross validations of different models for focus detection

Table 5.2: Detection of scope

	One Hot			Word Embedding		
Model	P	R	F-1	P	R	F-1
M1	0.824	0.802	0.812	0.822	0.786	0.803
M2	0.829	0.821	0.824	0.816	0.783	0.799
M3	0.860	0.847	0.853	0.848	0.811	0.828
M4	0.863	0.851	**0.857**	0.850	0.826	0.837
M5	0.697	0.551	0.614	0.700	0.565	0.625

Four of our models, i.e. all except M5, were able to predict the cues almost perfectly. Since our corpus consisted of limited number of unique cue words, the models were able to see those words most often during training. Among the five models, M5 differs with respect to the other models in its architecture.

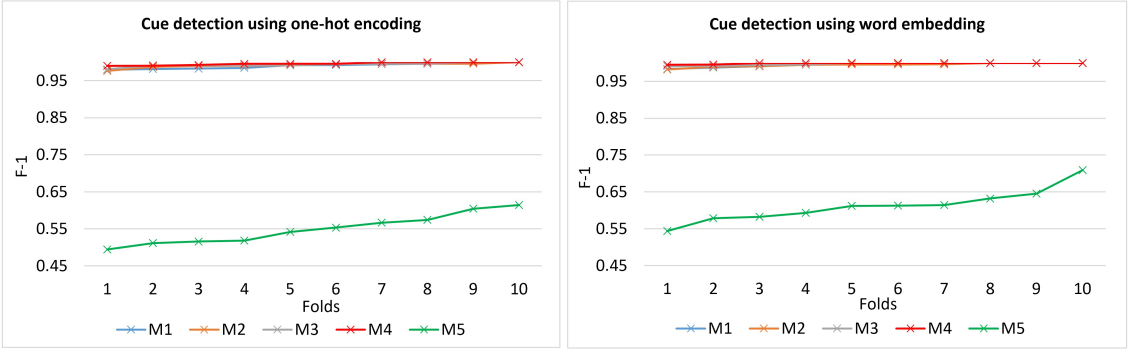

Fig. 5.5: F-1 scores of 10-folds cross validations of different models for cue detection

Table 5.3: Detection of focus

	One Hot			Word Embedding		
Model	P	R	F-1	P	R	F-1
M1	0.812	0.791	0.800	0.792	0.737	0.762
M2	0.819	0.788	0.802	0.818	0.747	0.779
M3	0.832	0.812	0.820	0.809	0.773	0.790
M4	0.839	0.843	**0.839**	0.817	0.780	0.797
M5	0.473	0.315	0.377	0.498	0.344	0.405

It is interesting to see that most of the models (except M5) trained with one-hot-encoding outperformed the model trained with word embedding for negation focus and scope detection. However. in the case of cue detection, models with word embeddings performed better. This could be explained by the fact that the word embedding vectors have been learned from a huge collection documents and our models were confused to some extent to generalize the DeepTutor data which is limited in size and diversity. whereas the In cae of M5, the absence of separate encoding and decoding phases could be the reason behind its poor performance as it had to predict the label immediately without seeing the full input sequence.

Figures 5.3, 5.4 and 5.5 show the F-1 scores (10-fold cross validation) for *scope*, *focus* and *cue* obtained with our models trained, respectively, with one-hot-encoding and word embeddings. From the figures, it can be seen that the the performance of all models except M5 remained consistent during each fold. Among the five models we presented, M4 performed best in average as well as in

Table 5.4: Detection of cue

	One Hot			Word Embedding		
Model	P	R	F-1	P	R	F-1
M1	0.994	0.985	0.990	0.996	0.994	0.995
M2	0.991	0.991	0.991	0.994	0.994	0.994
M3	0.991	0.995	0.993	0.998	0.996	0.997
M4	0.992	0.999	0.995	1.000	0.998	**0.999**
M5	0.621	0.493	0.549	0.707	0.542	0.612

individual run on 10-folds cross validation. The best performance of M4 is confirmed by its topmost position for the F-1 score plots of 10-folds cross validation. Also after through analysis of each run of 10-folds cross validation, we found that M4, had a best (among 10 different runs) F-1 score of 0.885 with and kappa of 0.822 for *focus* detection and F-1 of 0.889 with Cohen's Kappa of 0.840 for *scope* detection.

5.5 Conclusions

We explored various LSTM models and configurations for handling negation in tutorial dialogues. We have experimented with five models, broadly with two distinct network architecture, namely sequence to sequence tagger and tag sequence generator. We experimented and validated our models using real dialogues between student and an intelligent tutoring system. Our experiments suggests that the sequence to sequence tagger can handle well the substasks of negation scope, focus and cue detection, outperforming a previous method based on CRF that relied on human engineered features. A good choice of hyper-parameters of LSTM such as dropouts, number of units and layers could result in a competitive model for negation handling.

In this work, we have considered only the previous context while training the model. However, experiments has shown that LSTM RNNs trained with both past and future context perform better. In future work, we plan to build models that could be trained with both past and future contexts. We also plan to experiment

with a hybrid model that uses word embeddings as well as human engineered features.

Chapter 6

Markov Analysis of Learners' Conversation in Multi-Player System

In this paper, we conduct a Markov analysis of learners' professional skill development based on their conversations in virtual internships, an emerging category of learning systems characterized by the epistemic frame theory. This theory claims that professionals develop epistemic frames, or the network of skills, knowledge, identity, values, and epistemology (SKIVE) that are unique to that profession. Our goal here is to model individual students' development of epistemic frames as Markov processes and infer the stationary distribution of this process, i.e. of the SKIVE elements. Our analysis of a dataset from the engineering virtual internship Nephrotex showed that domain specific SKIVE elements have higher probability. Furthermore, while comparing the SKIVE stationary distributions of pairs of individual students and display the results as heat maps, we can identify students that play leadership or coordinator roles.

6.1 Introduction

In virtual internships students play the role of interns in a virtual training environment meant to simulate real internship experiences. The learning that occurs in virtual internships can be characterized by epistemic frame theory. This theory claims that professionals develop epistemic frames, or the network of skills, knowledge, identity, values, and epistemology (SKIVE elements) that are unique to that profession (Chesler et al., 2010). For example, engineers share ways of understanding and doing (knowledge and skills); beliefs about which problems are worth investigating (values), characteristics that define them as members of the profession (identity), and ways of justifying decisions (epistemology).

In this study, we propose a new method to characterize learners' professional skill development in virtual internships in terms of SKIVE elements distributions.

The basic idea is to use a Markov process approach to infer the stationary distribution of SKIVE elements based on an analysis of interns'/learners' conversations with other players, e.g. a mentor or intern, in engineering virtual internships. Specifically, we analyze interns' online conversations during the design process, a key activity in the engineering virtual internships such as Nephrotex (NTX) in which students research and create multiple engineering designs (Bagley & Shaffer, 2009).

Following prior work, elements of the engineering epistemic frame are operational zed as discourse codes in order to detect when students activate such SKIVE elements during conversations. An example of the identification of SKIVE elements as discourse codes in virtual internship conversations is shown in Table 6.1 where the student utterance encodes a reference to design Skills.

Table 6.1: An example of an utterance and SKIVE codes

Utterance	S	K	I	V	E
Let me know if you have any questions about their requirements for membrane design.	1	0	0	0	0

While an empirical distribution could be derived by computing the relative proportion of each activated SKIVE elements during conversations, our goal is to infer the true or stationary distribution of SKIVE elements for each student by modeling students' epistemic frames as Markov processes. The stationary distribution is the true distribution of SKIVE elements that would be observed if the student would talk forever (or an extremely long period of time). We designed Markov processes for SKIVE elements in virtual internships as briefly explained next. Markov processes are characterized by a set of states, which in our case are the SKIVE elements, and a set of transition probabilities, which we derive from analyzing the activation of SKIVE elements during the virtual internship conversations. For instance, we consider transitions from SKIVE elements activated

by a student in prior dialogue utterances to the SKIVE elements activated by the student in the current utterance. More precisely, we will use a moving window to delimit the number of previous dialogue utterances to consider when deriving the transitions. The size of the prior context moving window (in terms of number of utterances) can be set by the experimenter, as we will explain later. The larger the window the more likely we will identify transitions between various SKIVE elements, therefore, reducing data sparseness issues. On the other hand, a larger previous context window, i.e. one that includes many previous utterances, will account for long-distance transitions, i.e. between SKIVE elements activated in utterances that are far apart, which may be less relevant.

Given the above design, we experimented with several methods of deriving Markov processes to infer SKIVE elements' stationary distributions. For instance, we differentiated between methods that consider utterances from a single player versus all utterances (of all players). Also, we varied the way we derive the state-to-state transition counts, which are used to compute the transition probabilities, from a source state to a destination state: transitions between SKIVE elements/states in any utterance in the moving window will make the same contributions to the final transition count versus a penalizing model in which transitions from SKIVE elements/states farther away in the prior dialogue context are contributing less, e.g. there is a discounting parameter. We also compared models with and without a dummy SKIVE element/state (noSKIVE state) used to characterize utterances in which no SKIVE element is present (i.e., the student is not mentioning any discourse code indicative of a SKIVE element).

Once we inferred the SKIVE epistemic frame in terms of a stationary distribution for each student/intern, we compared students' SKIVE epistemic frames against each other and also against an average epistemic frame distribution obtained by computing an average of the stationary distributions of students'

epistemic frames. We compare the epistemic frame distributions using Kullback-Leibler (KL) divergence.

Our work has a merit in the sense that it provides a more rigorous way of describing students' emergence/mastery of SKIVE elements in terms of stationary/true distributions as opposed to empirically derived distributions. In the next sections, we discuss related work, Markov Chain theory and its use in virtual internship, the proposed conversation models, the engineering virtual internship Nephrotex datasets we used, and experiments and results. The paper ends with Conclusions and Future Work.

6.2 Related Works

The epistemic network analysis (ENA) framework was proposed as a way to characterize learning during internship when young apprentices are beginning their professional career by interacting with seasoned professionals (Bagley & Shaffer, 2009; D. W. Shaffer et al., 2009). ENA is grounded in epistemic frame hypothesis (D. W. Shaffer, 2006a) according to which professionals develop epistemic frames or the network of *skills*, *knowledge*, *identity*, *values*, and *epistemology*(SKIVE) that are unique to that profession. The network or interconnections between concepts enable the process of assessment of learning progression in context. The ENA framework offers evidence-centered design that provides evidence of learning by systematically linking models of understanding, observable actions, and evaluation rubrics.

Rupp and colleagues (Rupp et al., 2009) described a method to represent students' epistemic frames using ENA. In their method, the sequence of activities in the Urban Science, a virtual internship game, is divided into time slices and each slice is coded based on whether the slice activates one or more of the SKIVE codes. They then constructed an adjacency matrix of these codes for each slice based on whether any two of the codes co-occur in the slice. A cumulative adjacency matrix is also derived for a player/intern or the mentor from the whole sequence of activities

(an accumulation of all activated SKIVE codes over all time slices). They used different statistics such as overall weighted density, absolute and relative centrality and rescaled cumulative association to build a network structure of SKIVE elements based on the adjacency matrices. Students' mastery of SKIVE elements was measured by distance metrics among different players under comparison.

In another work, Bodin (Bodin, 2012) analyzed university physics students' epistemic frames while working on a task in which they were supposed to simulate a particle-spring model system. Students' epistemic frames were analyzed before and after the task using a network analysis approach derived from an analysis of interview transcripts. They found that students change their epistemic frames when switching from a modeling task to a physics task.

Zhu and Zhang (Zhu & Zhang, 2016) did a pilot investigation to understand the patterns of the communication and connections of engineering professional skills (EPSs). For the pattern of communication, they identified who was talking to whom at different points of time and identified whether one or more EPSs co-occurred in an utterance. They applied social network analysis to the resulting co-occurrence network and found that the high-performing group tend to show denser and more balanced network connections in both the communication and EPS networks.

In our work, we used Markov process theory to characterize students' development of SKIVE epistemic frames in terms of stationary distributions, as described next. That is, we consider students' development of SKIVE elements as a Markov process in which there is a state corresponding to each SKIVE element and the transitions among those SKIVE elements are observed during dialogues. Based on the transitions, we can derive the stationary distribution of an interns' SKIVE elements, which can be regarded as a reflection of that students' master of their target profession's skills, knowledge, identity, values, and epistemology.

6.3 Markov Process

A Markov process is a random process characterized by a set of states and transition probabilities among these states. The probability of the Markov process being in a particular state only depends on the previous state. The dependency of the current state only on the previous state, or a limited history of previous states, is called the Markov property of a Markov process. The Markov property is expressed formally using Equation 6.1, where X1, X2,..., Xn is a sequence of random variables.

$$P(X_{n+1}|X_1, X_2, ..., X_n) = P(X_{n+1}|X_n) \tag{6.1}$$

The transition probability matrix of a Markov process is of the form shown in Equation 6.2, where rows indicates the source state and columns indicate target/destination states. A particular element, e.g. p13, indicates the probability of making a transition from source state, say 1, to a destination state, say, 3.

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix} \tag{6.2}$$

The transition probabilities can be used to predict the probability of being in a particular state after a number of transitions when starting from a particular state. For instance, the probability of being in state j after 2 transitions/steps when starting from state i is shown in Equation 6.3, where superscript two (2) indicates the number of steps.

$$p_{ij}^{(2)} = p_{i1}p_{1j} + p_{i2}p_{2j} + ... + p_{in}p_{nj} \tag{6.3}$$

Furthermore, equation 2 is simply the dot product of the i$^{\text{th}}$ row and j$^{\text{th}}$

column vectors of the transition matrix. Therefore, we can predict the future state of the Markov process by obtaining the power of the transition matrix. The probability that a Markov process reaching state j from state i after transiting to k-1 states in between is given by $(i,j)^{th}$ element of the matrix $P^k = P*P*\ldots*P$ (P multiplied k times). Importantly for our work, the convergence theorem of Markov processes indicates that when k tends to infinity the matrix P^k attains a stationary state in which all rows are equal (Tierney, 1994).After reaching convergence, the probability of reaching a state is constant irrespective of the state from which the system had started. In our case, we rely on this convergence theorem to derive the stationary distribution of students' SKIVE elements based on transition probabilities derived from conversations during virtual internships, as explained next.

6.3.1 Markov Processes for Epistemic Frames

Besides content knowledge, students need to master their target profession's *skills*, *knowledge*, *identity*, *values*, and *epistemology* (SKIVE or epistemic frame elements). We propose here a novel way to monitor and assess students' mastery of the SKIVE elements in terms of stationary distributions of the states of a SKIVE Markov process in which there is a state for each of the SKIVE elements. We rely on students' activation of SKIVE elements during their conversations with other players in the virtual internship to characterize the underlying Markov process and infer the stationary SKIVE distribution for each intern.

For this purpose, every utterance of a conversation is being annotated with binary codes indicating whether a particular SKIVE element is present or absent in the utterance. That is, whether the student activated the corresponding SKIVE elements during his conversational moves.

We model each SKIVE element as a state of an underlying Markov process. Because a student can activate multiple SKIVE elements in the same utterance,

another option is to consider as a state a combination of SKIVE elements that students may articulate in any given utterance. We opted for the one-SKIVE-element per Markov process state option because we are interested in characterizing students in terms of SKIVE elements distributions as opposed to combinations of such elements. Secondly, considering all possible combinations SKIVE elements increases the complexity of the Markov process by increasing exponentially the number of states to all possible subsets of SKIVE elements, i.e. to 2^n states where n is the number of SKIVE elements. This will lead to two problems: (i) a data sparseness problem when deriving the transition probabilities and (ii) difficulty with interpreting the outcome.

We derive transition probabilities for each student's Markov process from the sequence of SKIVE elements identified in student's utterances during virtual internship conversations. That is, SKIVE elements activated in previous utterances are considered source states and SKIVE elements activated in the current utterance are considered target states of a state transition. We count each such transition from a source SKIVE state to a target SKIVE state and then normalize the values across all transitions with the same source state to infer the transition probabilities.

There are three important aspects of the way in which we derive the transition probabilities. First, instead of using the full previous dialogue context to detect source states for the transitions, we use a limited dialogue history in the form of a moving window of k previous utterances relative to the current utterance inspired from previous work (Rupp et al., 2009; Rus et al., 2014). The larger the window the more likely it is that we will identify transitions between various SKIVE elements, therefore, reducing data sparseness issues. On the other hand, a larger previous context window, i.e. one that includes many previous utterances, will account for long-distance transitions, i.e. between SKIVE elements activated in utterances that are far apart, which may be less relevant.

Second, when analyzing conversations to count the number of transitions from one SKIVE element to another, one can treat transitions from utterances close to each other the same way as transitions from utterances far apart in which case both such types of transitions will contribute an equal count of 1 to the final count. An alternative is to give less weight to transitions derived from utterances far apart. We present results with both these weighting methods.

A third key aspect with respect to deriving the transition probabilities is what utterances in a conversation to consider: the utterances of the student being analyzed or all utterances of all the players. We present results with both models: student-utterances vs. all-utterances.

A formal description of our conversation models and the transition probabilities are derived is presented next.

6.4 Conversation Model

A conversation is a sequence of utterances $u_1...u_T$ where an utterance u_k is coded with a set of binary codes corresponding to each of the SKIVE elements $C_{k1}...C_{kN}$. We also define a set of weights $w_1...w_N$ corresponding to an utterance u_k such that a weight w_j is given by:

$$w_j = \begin{cases} \sum_{p=k-n}^{p=k-1} \alpha^{p-k+1} C_{pj} \text{ if previous utterance window is weighted} \\ 1 \text{ otherwise if } C_{pj} = 1 \text{ for some p and unweighted window} \\ 0 \text{ otherwise if } C_{pj} = 0 \text{ for all p and unweighted window} \end{cases}$$

where α is a decay factor (set to 2, as explained later)that penalizes the contribution of farther utterances in the window/slice corresponding to the current utterance. C_{pj} is the j^{th} SKIVE element of previous utterance p within the moving window of size n.

The weighted count transition matrix M of dimension NxN is obtained by

the algorithm listed in Table 6.2. The above conversation model allows us to derive the transition probabilities among SKIVE elements by constructing an adjacency, binary or weighted, matrix.

Table 6.2: Algorithm to obtain weighted count matrix

Initialize: M = a zero matrix do for each utterance ui:
U = a NxN zero matrix
do for each code Cj of ui:
*uij =wj*Cj, where uij is an element of U*
M = M + U

The adjacency matrix for a given student is basically constructed by scanning her conversation utterances and counting the number of times the student activates SKIVE element B in the current utterance while activating SKIVE element A in one of the k previous utterances included in our moving window of size k. The adjacency matrix thus obtained cumulatively counts the number of transitions between any pair of SKIVE elements. The result is a state transition matrix whose entries are raw cumulative count of transitions from one SKIVE element to another.

The final state transition probability matrix is obtained by dividing each entry in the adjacency matrix by the sum of the elements in the corresponding row. In order to deal with sparse matrices, which have many zero entries, we do Laplace's add-one smoothing (Lidstone, 1920).

6.4.1 Adjacency Matrix with Single Player Window

In this model, an adjacency matrix is created for each player by considering only utterances of that player. For example in Table 6.3, for any player pl_i, the utterance for the player is the set of all utterances where pl_i is marked (x).

Table 6.3: Truth table of utterance and players in a conversation

	pl_1	pl_2	...	pl_n
utt_1	x			
utt_2		x		
...				
utt_k		x	...	

6.4.2 Adjacency Matrix with Multi-Player Window

In this model, an adjacency matrix is created for each player by taking into consideration utterances spoken by all the players in the conversation. In this case, in Table 6.3, for any player pl_i, the utterances for the player is the set of all the utterances utt_1 through utt_k.

6.5 The Nephrotex Dataset

We experimented with data from the virtual internships Nephrotex in which groups of students work together on a design problem, e.g. designing filtration membranes for hemodialysis machines, with the help of a mentor.

In the Nephrotex dataset, there are 25 players divided into five groups. Each group is assigned a virtual room to work together on a task.

Table 6.4: Excerpt of a conversation in Nephrotex; only few SKIVE codes(design, professional, collaboration and data) are shown

Player	Utterence	s.des	s.prof	s.coll	s.data
4	Let me know if you have any questions about their requirements for membrane design.	1	0	0	0
4	At Nephrotex, we have internal consultants who are experts in their fields.	0	0	0	0
18	When they say carbon nanotubes, to which surfac-tant are they referring to?	0	0	0	0
16	I believe it's any of them. It's just using the carbon nanotubes in general.	0	0	0	0

Once the task is finished, the players are assigned to other groups and a group is again assigned a room to work on a task. In total, there were 10 unique groups and 19 unique rooms formed. The dataset consists of a total of 2,970 utterances with an average of 37 utterances per room.

Table 6.4 is an excerpt of a conversation in Nephrotex. The utterances are coded with 20 SKIVE elements. While analyzing the dataset, we found that some of the utterances do not contain any SKIVE elements, hence a row attributed to that utterance has zero counts across all SKIVE elements, i.e. columns in Table 6.4. We handled such scenario following two different approaches. In a first approach, we discarded all the utterances with all-zero counts. In another approach, we introduced a dummy state, called no-SKIVE state, which indicates a state when no SKIVE element was activated by a student in an utterance.

6.6 Experiments and Results

We experimented with a combination of single-player vs multi-player models, weighted vs. non-weighted counts/binary counts, and Markov processes with or without no-SKIVE state. For the weighted window models, we selected a decay factor of $\alpha = 2$ thus penalizing by a factor of 2 transition counts from previous utterances for each one unit increase in distance from the current utterance.

For each model we ran the Markov process iteratively until it converged, i.e. reaching the stationary state. Figure 6.1 shows the average stationary distribution of SKIVE elements when the utterances from all the players are considered while Figure 6.2 shows the average stationary distribution derived using only utterances of one player.

Because the Nephrotex dataset is an engineering design internship, engineering specific components such as design, data, manufacturing, attributes, materials and engineers have higher probabilities compared to others. Also adding a no-SKIVE state in the analysis, shifted a good chunk of probability to the no-SKIVE state. This is the case because a significant part of the conversation consist of short dialogue turns in which the speakers use elliptical responses, in which much is implied from the context, or they focus on general conversation and process topics, e.g. greeting each other or asking about how to use the system.

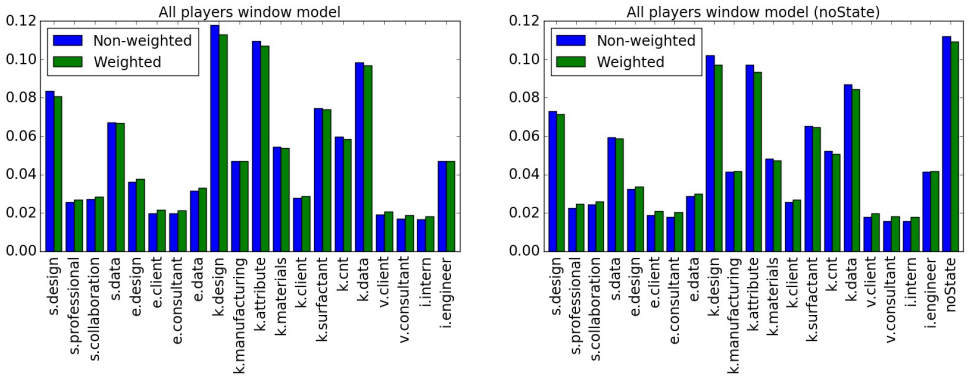

Fig. 6.1: Distribution of probabilities of SKIVE elements for all-player conversation model with non-weighted and weighted window and distributions when noState added

When comparing the weighted window model to its nonweighted counterpart, the distributions are similar as confirmed by distance measures (KL-divergence) between the corresponding distributions of SKIVE elements of the weighted and non-weighted models (KL=0.00031 for all player without no-SKIVE state; KL=0.00035 for one player without no-SKIVE state). However, one can notice that the probability distribution of the least frequent SKIVE elements are boosted. This is in a way a desired effect because the most frequent elements, if not penalized, tend to dominate by the simple fact that they occur in more utterances throughout a conversation and therefore are more likely to be present in a moving window which in turn leads to increased transition counts. The weighted model penalizes frequent components when occurring in remote utterances relative to the current utterance.

Once the stationary distributions of SKIVE elements were obtained, we conducted an analysis of students' SKIVE profiles by computing KL-divergence (KL) scores between pairs of distributions of SKIVE elements for individual students. A summary of the KL scores is shown as a heat map in Figure 3. Student players are sorted based on their average KL score with other players. The left vertical color bar in the maps show the intensity of the user's average KL score in sorted order.

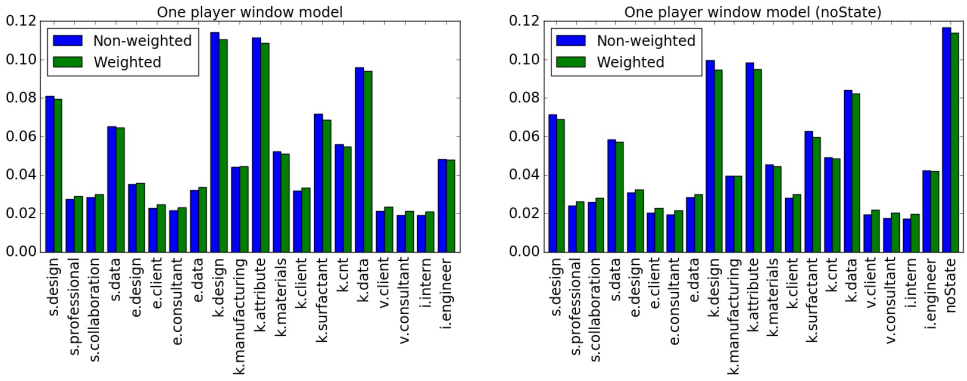

Fig. 6.2: Distribution probabilities of SKIVE elements for single player conversation model with non-weighted and weighted window and distributions when noState added

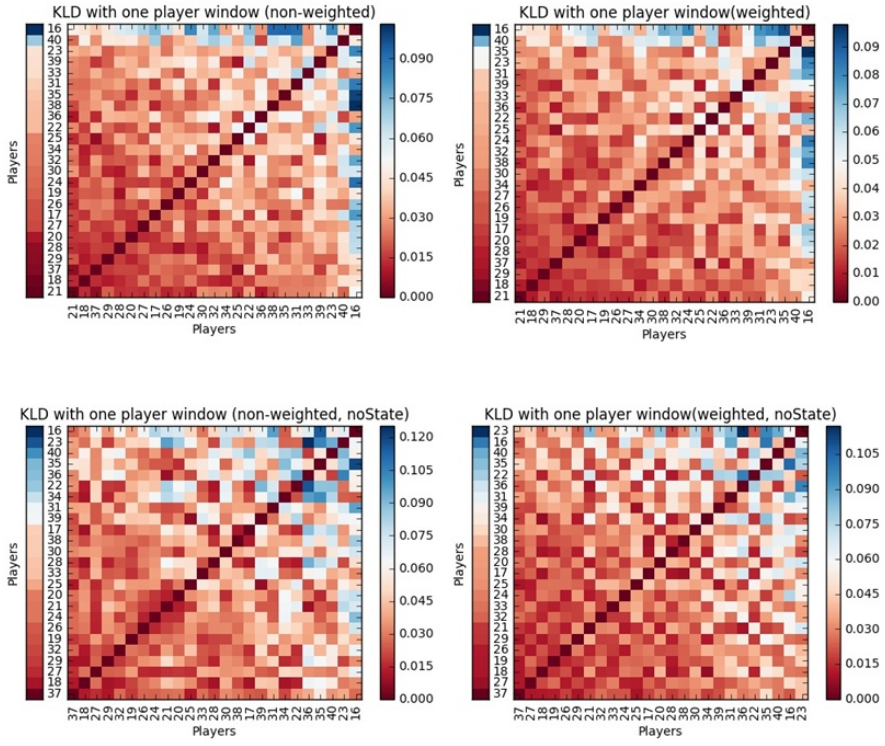

Fig. 6.3: KL-divergence of distribution SKIVE elements of players for window with single player utterances (bottom two are for noState)

It can be seen that some players have similar distributions, e.g. those shown in the lower left corner of the heat maps in Figure 6.3. The lower left corner corresponds to lower divergence scores. When using a no-SKIVE state and weights, the distributions of SKIVE elements between players seem to be more similar as shown in the heat maps on the lower right hand side of Figure 6.3. Furthermore, adding a no-SKIVE state (bottom left and bottom right) revealed that some of the players move from an upper position, corresponding to a higher average divergence score, to lower-divergence positions in the heat map. Those that move to lower-divergence positions are more likely to have utterances in which no SKIVE elements are activated. They may correspond to students playing more leadership/coordinator roles as their utterances focus more on process and conversational management topics and less on SKIVE elements. We only show results for single-player utterances models due to space reasons.

6.7 Conclusions

We conducted a Markov process analysis of students' mastery of epistemic frames which is generally applicable to any epistemic frame. We have experimented and validated our method on data from an engineering epistemic frame using eight different ways to model Markov processes for each student participating in engineering virtual internships. The comparison of the distribution of SKIVE elements for individual students in models with noState revealed that some students may play more managerial or coordinator roles than others.

In future work, we plan to use the stationary distribution of SKIVE components obtained from this analysis to better understand students' effectiveness of acquiring much needed skills to be successful professionally. Furthermore, we plan to develop a dual Markov process to infer stationary distributions of states and transitions.

Chapter 7

Speech Act Categorization in Multi-Player Conversational Systems

This work is a step towards full automation of auto-mentoring processes in multi-player online environments such as virtual internships. We focus on automatically identifying speaker's intentions, i.e. the speech acts of chat utterances, in such virtual internships. Particularly, we explore several machine learning methods to categorize speech acts, with promising results. A novel approach based on pre-training a neural network on a large set of (and noisy) labeled data and then on expert-labeled data led to best results. The proposed methods can help understand patterns of conversations among players in virtual internships which in turn could inform refinements of the design of such learning environments and ultimately the development of virtual mentors that would be able to monitor and scaffold students' learning, i.e., the acquisition of specific professional skills in this case.

7.1 Introduction

In virtual Internships, interns gain professional experience without actually being present in a physical, actual company. While working as interns, the students participate in activities such as solving designated problems or tasks for which they actively interact with their mentor(s) as well as other interns through instant text messages, voice messages, chatrooms, and multimedia elements. The learning that occurs in engineering virtual internships, our focus, can be characterized by epistemic frame theory. This theory claims that professionals develop epistemic frames, or the network of skills, knowledge, identity, values, and epistemology that are unique to that profession (D. W. Shaffer, 2006b). For example, engineers share ways of understanding and doing (knowledge and skills); beliefs about which problems are worth investigating (values), characteristics that define them as

members of the profession (identity), and a ways of justifying decisions (epistemology).

It is important to understand patterns of conversations between the various players in a virtual internship in order to refine the design of such virtual internships and to ultimately develop a virtual mentor that would be able to monitor and scaffold students' learning, i.e., the acquisition of specific professional skills in this case. Currently, virtual internship environments rely on human mentors. Our work here is a step towards a deeper understanding and full automation of the mentoring process. Indeed, understanding the mentoring process implies detecting patterns of actions by the mentor and by the students that are effective. Since conversations are the main type of interactions between the mentors and the student interns, understanding the actions or intents behind each utterance in the conversations is critical. We offer here such solutions to automatically detecting the intent, or speech act, behind chat utterances in virtual internships. Furthermore, such solution are critical to fully automate the mentoring process, i.e., to building auto-mentors. Indeed, knowing students' speech acts can inform an automated mentoring agent to plan the best reply. For instance, if a student is greeting, the system should respond with a greeting or if a student is asking a question the system should plan to, for instance, answer the question.

Speech acts are a construct in linguistics and the philosophy of language that refers to the way natural language performs actions in human-to-human language interactions, such as dialogues. Speech act theory was developed based on the "language as action" assumption. The basic idea is that behind every utterance there is an underlying speaker intent, called the speech act. For instance, the utterance "Hello, John!" corresponds to a greeting, that is, the speaker's intention is to greet, whereas the utterance "Which web browser are you using?" is about asking a question. As already hinted earlier, discovering learners' patterns of actions in the

form of patterns of (speech) acts in virtual internships could be revealing. For instance, we may find that interns that ask more questions acquire better and faster target professional skills based on the theory that asking more relevant questions indicates a more active and engaged learner which typically leads to more effective and efficient learning processes.

Labeling utterances with speech acts requires both an analysis of the utterance itself, e.g., "Hello" clearly indicates a greeting, but also accounting for the previous context, i.e., previous utterances in the conversation. For instance, after a question, a response most likely follows. This pattern holds in dialogues, i.e., interactions between two conversational partners where there is a clear pattern of turn-taking; that is, a speaker's turn is followed by a turn by the other speaker. However, in multi-player conversations such as the one that we deal with in this work, identifying the previous utterance that is most relevant to the current one is more difficult. For example, in the snapshot of conversation shown in Table 7.1 from one of our virtual internships, the question in chat utterance 3 from *player2* is addressed to the *mentor* whose reply is in utterance 6. The next *Player2's* reply is in utterance 9. Indeed, in such multi-party conversations, it becomes more challenging to link a target utterance to the previous one that triggered it. The complexity of untangling such multi-player conversations is further increased as the number of participants increases. Therefore, even though the speech act of an utterance is determined to some degree by the previous, related chat utterances, in this work we explore a method for speech act classification that relies only on the content of the target utterance itself, ignoring the previous context.

To this end, we used various existing classifiers such as Naive Bayes and decision trees along with a Neural Network (NN) approach. Based on previous experience such as (Samei, Li, Keshtkar, Rus, & Graesser, 2014; Rus, Moldovan, Niraula, & Graesser, 2012), we selected leading words in each utterrance as the

Table 7.1: A snapshot of conversation in nephrotex

S.N.	Speaker	Utterance
1	mentor	I'm here to help you.
2	player1	hi!
3	player2	*Has anyone been able to get the tutorial notebook to open?*
4	player3	Hey
5	player4	Hello!
6	mentor	*Which web browser are you using?*
7	player3	are you guys real?
8	player1	yes we're real lol
9	player2	*I switched to Firefox, now everything is working. Thanks!*

features of the underlying model. The feature-based representations of utterances were then fed into Naive Bayes and decision tree classifiers. For neural networks, we used the pre-trained sent2vec (Pagliardini, Gupta, & Jaggi, 2017) model, trained on a large collection of Wikipedia articles, to map an entire utterance onto a vector representation or embedding. Nevertheless, our data is dialogue data which differs from Wikipedia texts to some degree. To compensate for this discrepancy, the basic model is used to further train a small neural network using a comparatively small domain specific dataset in order to improve the predictive power for the type of instances seen in our dataset. That is, this is a form of transfer learning where our model first uses generic knowledge from the pre-trained Wikipedia model which is then tranferred or adapted to a specific domain data by training with domain data. Furthermore, using pre-trained models can also lead to better parameter learning in NN (Pan & Yang, 2010).

We also investigated a novel approach to building a speech act classifier for multi-player conversational systems using a mix of noisy and golden data, as explained next. In this approach, we trained a decision tree model with a small set of human annotated data and then used that trained model to generate (noisy)

labels for a much larger collection of utterances. The noisy labeled utterances were then used to pre-train the neural network and then further trained with the human annotated gold dataset. The advantage of pre-training here is to have a huge collection of training data to pre-train the network and then refine the training using the (smaller) human-annotated (noise-free or gold) dataset.

Next, we present a quick overview of related work in this area before presenting details of our methods and experiments and resuts.

7.2 Background

As mentioned, our approach to label utterances with speech acts is based on the speech act theory according to which when we say something we do something (Austin, 1975; Searle, 1969). Austin theorized the acts performed by natural language utterances. Later on, Searle (Searle, 1969) refined Austin's idea of speech acts by emphasizing the psychological interpretation based on beliefs or intentions. According to Searle, there are three levels of actions carried by language in parallel. First, there is the locutionary act which consists of the actual utterance and its exterior meaning. Second, there is the illocutionary act, which is the real intended meaning of the utterance, its semantic force. Third, there is the perlocutionary act which is the practical effect of the utterance, such as persuading and encouraging. In a few words, the locutionary act is the act of saying something, the illocutionary act is an act performed in saying something, and the perlocutionary act is an actperformed by saying something. For example, the phrase "Don't go into the water" might be interpreted at the three act levels in the following way: the locutionary level is the utterance itself, the morphologically and syntactically correct usage of a sequence of words; the illocutionary level is the act of warning about the possible dangers of going into the water; finally, the perlocutionary level is the actual persuasion, if any, performed on the hearers of the message, to not go into the water.

Many researchers have explored the task of automatically classifying speech acts as well as the related task of discovering speech acts. For instance, Rus and colleagues (Rus et al., 2012) proposed a method to automatically discover speech act categories in dialogues by clustering utterances spoken by participants in educational games. In our case, we use a predefined taxonomy of speech acts which was inspired by Rus and colleagues' work and further refined by dialogue experts.

The same group of researchers explored the role of Hidden Markov Models (HMMs), a generative model, and Conditional Random Fields (CRFs), a discriminative model, in classifying speech acts in one to one human tutorial sessions (Rus et al., n.d.). They demonstrated that the CRF model with features constructed from the first three tokens and last token of previous, next and current utterances, length of current utterance, and other surface features such as bigrams and the speech acts of context utterances performed better than HMM models. They have not worked with multi-party conversations as it is the case in our work.

In other work, Moldovan and colleagues (Moldovan, Rus, & Graesser, 2011) applied supervised machine learning methods to automatically classify chats in an online chat corpus. The corpus consisted of online chat sessions in English between speakers of different ages. Their supervised approach relied on an expert defined set of speech act categories. In their work, they hypothesized that the first few tokens were good predictors of chat's speech act. Samei et al. (Samei et al., 2014) adopted Moldovan and colleagues' hypothesis about the predictive power of first few tokens and extended the supervised machine learning model with contextual information, i.e., previous and following utterances. From their experiments with data from an online collaborative learning game, they found that the role of context is minor and therefore context is not that important and can mostly be ignored in predicting speech acts. Similar to those works, we also explore the effectiveness of leading word tokens in utterances for Naive Bayes and decision tree based classifiers.

Ezen and Boyer (Ezen-Can & Boyer, 2013) proposed an unsupervised method for dialogue act classification. They used a corpus from a collaborative learning programming tutor project which consisted of dialogues between pairs of tutors and students collaborating on the task of solving a programming problem. They applied an information retrieval approach in which the target utterance was considered as a query and the rest of the utterances were considered as documents. Based on the ranked list of relevant utterances to the query utterance, a vector representation is derived for each query utterance. The vector representation is then fed into a k-means clustering algorithm to identify clusters of utterances. For evaluation purposes, they used manually labeled data. Each cluster was assigned the majority human-generated label of all utterances in the cluster. An utterance that was placed in a particular cluster by the k-means clustering algorithm was assigned the label of that cluster as its speech act category for evaluation purposes. It should be noted that they varied the number of clusters to obtain a maximum overall accuracy of the discovered labels. Their algorithm outperformed a previous approach for dialogue act clustering, which Ezen and Boyer used for classification and which relied on a simple tf-idf representation and cosine similarity for clustering.

Kim and colleagues investigated the task of classifying dialogue acts in multi-party chats (S. N. Kim, Cavedon, & Baldwin, 2012). They analyzed two different types of live chats: (i) live forum chats with multiple participants from the US Library of Congress and (ii) Naval Postgraduate School (NPS) casual chats (Forsyth, 2007). In order to classify the utterances in the chats in various speech act categories, Kim and colleagues (J. Kim, Chern, Feng, Shaw, & Hovy, 2006) used speech act patterns which they defined manually using cue words derived from the utterances. They classified the discussion contributions into six speech act categories. They found that the previous chat utterances used as context did not contribute significantly to predicting speech acts in multi-party conversations until

the entanglement amongst the utterances was resolved. Our work is similar to theirs in the sense that we analyze multi-party conversations. Nevertheless, our work is conducted in the context of the virtual internship Nephrotex, where learners focus on specific design problems as opposed to the types of conversations used by Kim and colleagues such as the casual NPS chats, which did not focus on a particular given task. We do not explore the accuracy of our methods in context. Furthermore, we do not resolve the entangled dialogues and then use contextual information for speech act classification. We do plan to address the role of context and entanglement in multi-party conversations in future work.

A regular expression based speech act classifier was proposed by Olney et al (Olney et al., 2003). Their classifier used regular expression which they called a finite state transducer to classify utterances of AutoTutor, an intelligent tutoring system. They showed that the classifier constructed by cascading parts of speech information, the finite state transducer, and word sense disambiguation rules yielded good performance in classifying utterances into 18 categories. We have not compared our work with a regular expression based classifier due to the labor intensive aspects of such an approach. Typically, such regular-expression approaches should lead to high-precision results and not generalize very well unless they target speech act categories which are more or less closed-class such as greeting expressions (there is a limited number of expressions in which someone can greet).

7.3 Engineering Virtual Internships: an Overview

Our work presented here was conducted on conversations among students and mentors in *Nephrotex* (NTX), a virtual internship. Nephrotex was designed and created to improve engineering undergraduate students' professional skills. It was incorporated into first-year engineering undergraduate courses at the University of Wisconsin-Madison (D'Angelo, Arastoopour, Chesler, Shaffer, et al., 2011).

In NTX, groups of students work together on a design problem, e.g.

designing filtration membranes for hemodialysis machines, with the help of a mentor. Working on a design problem involves choosing design specifications from a set of input categories. Each student is assigned to a team of five members. There were five such teams who were each expected to learn about one of five different materials.

After completing a set of preliminary tasks, students design five prototypes to submit for testing. Later, they receive performance results for these prototypes which they have to analyze and interpret. Overall, students in each internship complete two such cycles of designing, testing, and analysis before deciding on a final design to recommend. During these cycles, students hold team meetings via the virtual internship's chat interface in which they reflect on their design process and make decisions on how to move forward. Once teams recommend a final design, they present this design to their peers. The conversations among the participants take place virtually via an online chat interface in Nephrotex, or in person outside of the class.

As previously mentioned, in this work, we focus on analyzing chat utterances in Nephrotex in order to discover the underlying speech act. Automated speech acts classification could have significant impact on scaling virtual internships to all students, anytime, anywhere via Internet-connected devices. This is not currently possible because the human mentors can only handle that much.

7.4 Methods

Our approach to classifying learner utterances in virtual internships relies on machine learning algorithms that take as input utterances represented in a feature space. The features in our case are either surface features (such as leading words) or latent features (such as dimensions in neural sentence embeddings). We developed and compared the performance of two different categories of classifiers that rely on these two types of representations. We describe briefly those classifiers, the features

we used, and the results obtained during experiments meant to validate the proposed classifiers.

7.4.1 Classifier Using Surface Features

The surface feature representation of a text uses a number of important lexical and syntactic elements such as leading words or the punctuation mark at the end of the utterance, e.g., the ending question mark at the end of a question. In conversation data such as chat utterances in virtual internship, lexical features such as leading words alone have competitive power in terms of speech act representation of the utterance. Therefore we adopted the model representation proposed previously (Moldovan et al., 2011; Rus et al., 2012) due to its solid theoretical foundations and competitive results. The basis of this approach is that humans infer speakers' intention after hearing only few of the leading words of an utterance. One argument in favor of this assumption is the evidence that hearers start responding immediately (within milliseconds) or sometimes before speakers finish their utterances (Jurafsky & Martin, 2009). Accordingly, we selected few leading words (first few words) of the utterance as the features to represent the utterance. Although we have experimented with different number of leading words, we report here results with the six leading words (first six words) as this combination yielded best performance as explained later. Once each utterance was mapped onto such a feature-representation, we performed experiments with two different types of classifiers: naive Bayes and decision trees.

Before feature construction, we pre-processed the utterances by lemmatizing the words and removed the punctuations. Although some of the punctuations, such as "*question mark (?)*" or "*exclamation mark (!)*", are predictive on some of the speech acts, they seem to not always be present in or seem to appear at improper places in the utterance. Hence we ignored the punctuations for our analysis here.

Table 7.2: Speech act taxonomy with examples

Speech Acts	Examples
expressive evaluation (XPE)	–It is excellent in all values except for cost –great –The lag is pretty bad
greeting (GRE)	–Welcome back interns ! –Hello Team !
metastatements (MST)	–sorry littles confused here –Whoops , I was reading that wrong . –lol
other (OTH)	–or addition –etc
question (QUE)	–Is biocompatability cummulative ? –who is going to write the email ?
reaction (REA)	–I 'm ok with this –alright , i think i agree with u guys
request (REQ)	–Please keep that in mind during your team selection of membrane prototypes . –K , I would like to start the team meeting now .
statement (STM)	–I read an article that said most dialyzers take 6 hours to run . –I can start the meeting with jamon ...

7.4.2 Classifier Using Latent Features

The other category of classifiers we used relies on latent features that were automatically learned using neural networks. These features are the components of automatically generated vectors that represent sentences. Such neural network generated vectors are derived from textual units such as character, letter n-grams, words and words n-grams. In our model, we adopted sent2vec, a sentence representation model proposed by Pagliardini and colleagues (Bojanowski, Grave, Joulin, & Mikolov, 2016; Pagliardini et al., 2017) and which was developed by training a neural network on a collection of Wikipedia articles.

Based on such latent representations of utterances, we designed a neural network model in two stages. First, the model obtained a latent representation for an utterance using the generic pre-trained sent2vec model. In a second stage, the embedded vector representation is used to further train our neural network to perform speech act classification.

While training the neural network with domain specific data, we applied two methods of training. In the first method, we used a small set of human annotated gold data for training and validation. In the second method, we pre-trained the neural network with noisy labeled data generated from a domain corpus and then further trained and validated the model with gold data. We will discuss in detail the process of generating noisy labels in the next section.

7.5 Experiments and Results

In this section, we present the experiments that were conducted and the results obtained, starting with a brief description of the data we used.

7.5.1 The Virtual Internship Conversation Dataset

Our dataset consists of a collection of more than 22 thousands utterances from the Nephrotex virtual internship. The eight categories of speech acts we used

are presented in Table 7.2 (acronyms are shown in parentheses) together with example utterances.

Table 7.3: Distribution of speech acts in corpus

	Human Labeled		Noisy Labeled	
Speech act	#	%Dist	#	%Dist
XPE	24	2.4	256	1.26
GRE	14	1.4	285	1.40
MST	40	4.0	405	2.00
OTH	11	1.1	166	0.82
QUE	173	17.3	3098	15.25
REA	202	20.2	3347	16.47
REQ	56	5.6	1041	5.12
STM	480	48.0	11719	57.68
Total	1000		20317	

From the examples, it could be observed that the leading tokens in each utterance are indicative of the underlying speech act shown in the first column. For instance, *greetings* start with "Hello" and "Welcome back" whereas *questions* start with wh-words ("Who") or auxiliary verbs ("Is") while requests start with "Please", which is typically used to ask for something in a nice manner.

The Data Annotation Process

Of the 22,317 utterances, 2,000 utterances were manually annotated by three annotators. Out of these 2,000 utterances, 1,000 utterances were used for training the annotators. Agreement among annotators was computed as the average of Cohen's kappa between all possible pairs of annotators. The average agreement between any two annotators was 0.64.

The remaining 1,000 utterances were labeled by the annotators after finishing their training. The average agreement, measured as Cohen's kappa, among the coders was 0.69. To generate a final, unique label for each annotated utterance in cases in which there were any disagreements, a discussion among the annotators

took place as well as the group of co-workers in the project team. We used the 1,000 human-labeled utterances as a gold dataset on which a 10-fold cross validation evaluation methodology was applied to evaluate the proposed speech act classification methods.

The Noisy Label Generation

The rest of the utterances in the whole dataset of 22,317 utterances was automatically labeled using the decision tree model trained on the first 1,000 instances labeled by trainee annotators. We chose decision trees to generate noisy labels because decision trees performed better than the Naive Bayes classifier. It should be noted that we used the other 1000 human-labeled gold data for 10 folds cross validations of our classifier models. Table 7.3 shows the distribution of speech acts in the gold and noisy labeled datasets. From the table, we observe that the noisy labels generated follow roughly comparable pattern of distribution for the speech acts that are more frequent in corpus. Therefore it makes sense to some extent to use those noisy labels to pre-train the neural network model.

7.5.2 Results

The results of the 10-fold cross-validation evaluation are summarized in Table 7.4 and Table 7.5. We report performance in terms of precision, recall, F-1 score, accuracy, and kappa. The data in Table 7.4 suggests that the performance of the neural network classifier is highest of all with an average F-1 score and accuracy of 0.764 and 0.779, respectively, and kappa of 0.666, which are the highest among all three types of classifiers including Naive Bayes and decision trees. Moreover, the two sample t-test on 10-fold cross validation accuracies revealed that, neural network performed significantly better than Naive Bayes ($p - value \approx 0.00$) and decision tree with ($p - value \approx 0.00$).

The results shown in Table 7.5 shows that the neural network model pre-trained with noisy labels improved the performance. The overall improvement

Table 7.4: Performance of naive bayes, decision tree and neural network classifiers

	NB			DT			NN		
SA	P	R	F1	P	R	F1	P	R	F1
XPE	0.200	0.042	0.069	0.000	0.000	0.000	0.556	0.208	0.303
GRE	1.000	0.143	0.250	0.000	0.000	0.000	0.667	0.143	0.235
MST	0.000	0.000	0.000	0.283	0.375	0.323	0.692	0.450	0.545
OTH	0.099	1.000	0.180	0.176	0.273	0.214	1.000	0.091	0.167
QUE	0.000	0.000	0.000	0.429	0.468	0.448	0.921	0.879	0.899
REA	0.354	0.342	0.348	0.630	0.599	0.614	0.687	0.683	0.685
REQ	0.000	0.000	0.000	0.143	0.143	0.143	0.614	0.482	0.540
STM	0.581	0.831	0.684	0.680	0.642	0.660	0.791	0.908	0.846
Wt. Avg.	0.370	0.482	0.406	0.549	0.536	0.542	**0.774**	**0.779**	**0.764**
	Accuracy = 0.482			Accuracy = 0.536			Accuracy = **0.779**		
	Kappa = 0.177			Kappa = 0.341			Kappa = 0.666		

in precision, recall, F-1 score, and accuracy is about 2% with about 2% better kappa when compared to the neural network classifier (Table 7.4) without using the much larger, noisy label dataset. However, a t-test showed that the accuracy of the noisy label trained neural network is not significantly better than neural network trained without noisy label data ($p - value \approx 0.53$). This could have happened because of the small samples used for the t-test: 10 from 10-folds cross validations. Using a larger number of folds, say, 50, could help us getting a large sample of accuracy values.

Table 7.5: Performance of noise label trained neural network classifier

SA	P	R	F1
XPE	0.900	0.375	0.529
GRE	1.000	0.357	0.526
MST	0.947	0.450	0.610
OTH	0.000	0.000	0.000
QUE	0.921	0.879	0.899
REA	0.774	0.713	0.742
REQ	0.742	0.411	0.529
STM	0.762	0.925	0.835
Wt. Avg.	**0.796**	**0.795**	**0.781**
Accuracy		**0.795**	
Kappa		0.685	

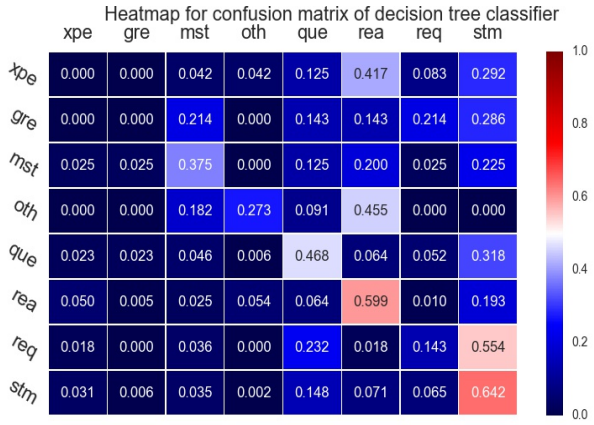

Fig. 7.1: Confusion matrix for classification of decision tree (values refer to percentage expressed in decimal, acronyms refer to the speech acts defined in Table 7.2)

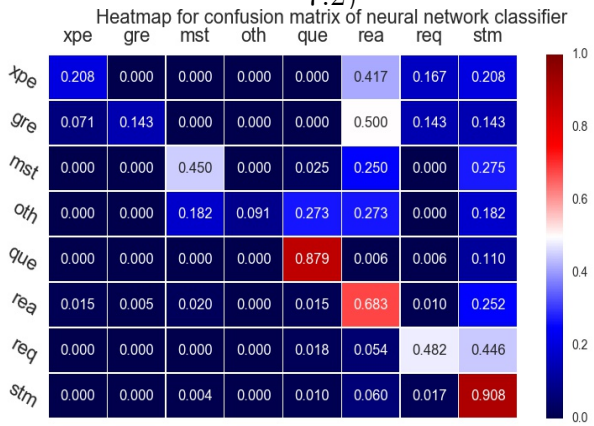

Fig. 7.2: Confusion matrix for classification of neural network (values refer to percentage expressed in decimal, acronyms refer to the speech acts defined in Table 7.2)

It can be observed from the table that the performance for the "*other*" category is the weakest among all four classifiers. The reason is because of the nature of those utterances which contain only a few tokens, i.e., one or two words (see Table 7.2), with a lot of variation in terms of lexical content. In addition, the human labeled dataset contained few instances for this category which resulted in poor performance when the neural network model was trained using the human labeled data. Similarly, in the noisy, automatically-labeled dataset there are many misclassified "*other*" instances which led to poor training of the neural network

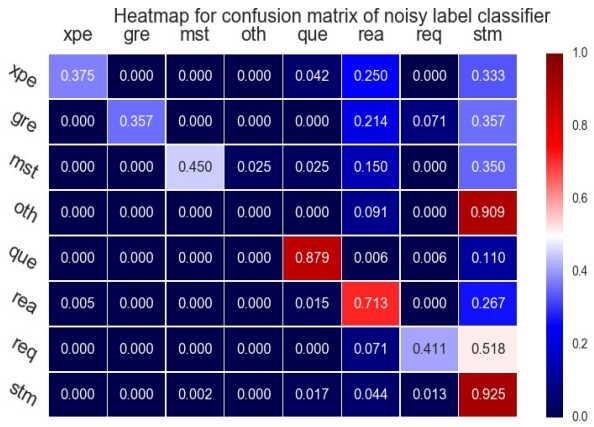

Fig. 7.3: Confusion matrix for classification of noise label trained neural network (values refer to percentage expressed in decimal, acronyms refer to the speech acts defined in Table 7.2)

model. Furthermore, the next phase of training the pre-trained neural network model with the human labeled data did not compensate enough because there were not sufficient "other" instances in the human labeled data to correct the pre-trained model. This is further supported by analyzing the confusion matrix where the number of true positives for the "*other*" category is 0%; the "*other*" category is labeled as "*statement*" 90% of the time in the case of the neural network model pre-trained with noisy labels (see Figure 7.3). Further evidence for this is provided by analyzing the confusion matrix for neural network trained only with gold labels where true positives for "*other*" utterances was 9% (see Figure 7.2). In this case, "other" utterances were labeled as "*question*" and "*reaction*". Other challenging speech acts are "*request*", which is most often confused with "*statement*". This is not surprising as the lexical composition of requests and statements is similar to some degree.

For decision trees, a quick analysis of the confusion matrix (see Figure 7.1) revealed that the true positives for "*expressive evaluation*" was 0%, being confused mostly with "*reaction*" or "*statement*" (41% and 29% of the time, respectively). Also, "*greeting*" is confused with "*metastatement*" by 21%, "*request*" by 21%, and "*statement* by 28%".

7.6 Conclusions

In this work, we explored several methods for speech act classification. We explored various classifier models with different categories of features as well as training strategies. We found that the latent features generated by a pre-trained sentence embeddings model (derived from a large Wikipedia corpus) yielded better performance compared to the other models. Besides that, the predictive power of the neural network model was further boosted when pre-trained with noisy label before training with expert-annotated data.

In future work, we plan to expand the current models by using more contextual information. Given the multi-party nature of our conversation data, before we can use contextual information, it is necessary to disentangle the conversations into sets of related utterances. Our future models will disentagle the multi-party conversations before attempting to use contextual information for speech act classification.

Chapter 8

Answer Assessment in Intelligent Tutoring System

In this chapter we present a noble method of assessing student answers in dialogue based intelligent tutoring systems. We focus to develop and evaluate our model for DeepTutor, an intelligent tutoring system for high school physics students. Particularly, we have experimented with different assessment models that are trained with the features generated from knowledge graph embeddings. Our experiment showed that the model trained with the feature vectors generated by Neural Tensor Networks (NTN) that are trained with combination of domain specific and domain general triplets perform better than the existing system.

8.1 Introduction

ITSs mimic human tutor in terms of conversational and pedagogical capabilities. Though they have been designed differently for different learning systems, the goal of these systems is to provide customized instruction or feedback to individual learner in order to maximize learning for the learner. As observed by several researchers (Nwana, 1990; Freedman, Ali, & McRoy, 2000; Nkambou, Mizoguchi, & Bourdeau, 2010), intelligent tutoring systems consist of four basic components: the domain model, the student model, the tutoring model and the user interface. The domain experts build domain model based on what to teach to the learners. While building the model, the domain experts apply the theory of learning which takes into account of concepts, rules and problem solving strategies of the domain to be learned.

The student model monitors student cognitive states and learning as the learning process advances. It uses learner's personalized data such as knowledge state and error made during learning, in order to provide personalized feedback and hints to guide student through the learning path that is designed in domain model.

Using information from domain and student model, the tutor model makes choice of tutoring strategies. The model makes instructional decision such as correct choice of teaching methods that suit best for individual learner. Moreover, it assesses the cognitive state of the learner and decides whether a learner can proceed to the next learning stage or revised the previous stage.

The user interface integrates the previous three models and provides environment through which the learner can interact with the system.

While an ITS has those components, the behavior of ITS during tutoring can be described using Van Lehn's (Vanlehn, 2006) two-loop framework: the outer loop and the inner loop.

The outer loop determines the appropriate ordering of tasks that are presented to the learners to work on.

The inner loop, on the other hand assesses learner's input and provide adaptive feedback while the learner is working on a task. During the process, the tutor asks a question and the learner submits answer to the question. The ITS monitors the learner's performance through its assessment model, updates its student model and determine whether the learner's answers has the expected concepts. If answer has all the expected concepts, the outer loop activates and selects next task. Otherwise, the inner loop uses its updated student model to provide appropriate feedback to get expected local goal.

Natural language understanding is the foundation of assessment model of ITS. Typically, automatic answer assessment systems assess student responses by measuring how much the student answer contains the targeted concept. These targeted concepts are created by subject matter experts and the semantic similarity between student's answer and target(reference) answer is measured to evaluate the correctness of the answer. The method to compute similarity scores by knowledge based approach such as using WordNet or corpus based approach such as Latent

Semantic Analysis (LSA) (Landauer, 2006), and Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003) have been popular from several years. Also several works (Pérez, Gliozzo, et al., 2005; Mohler, Bunescu, & Mihalcea, 2011) to harness the power of corpus driven approach with knowledge driven approach or vice versa have been studied in the past years.

However, the limitation of similarity based method is that such approaches assume student answer and reference are self contained. But most often, the student responses are indirect, elliptical or depends on context of domain. For example, in Table 8.1, the student answer A1 is more complete and semantic similarity approach could provide high similarity score with expected answer (E). However, previous study by Penumatsa et al. (Penumatsa et al., 2006) showed longer text yield larger cosine score. This means even student does not contain expected content but reference answers and student answers are longer, then similarity score would be high. On the other hand the correct short answer could be elliptical (A2 and A3) and give low similarity score with the expected answer. The small score will be even more prominent if some of the concepts are mentioned indirectly (for instance: "*downward force from the earth*" in A3). The concepts that are mentioned directly could be identified using semantic similarity approach by comparing definite sized chunk of text from student answer and reference answer, however for indirect mentions, it is more challenging.

We propose a knowledge graph based approach by representing the concepts in student answers and reference answer with embedding vectors that are learned directly from a knowledge graph which encodes indirect relationship between the concepts.

Table 8.1: An example of student tutor conversation in DeepTutor

Q: *What forces are acting on the puck while the puck is moving on the ice between the two players?*
A1: *The forces acting are gravitational force and the normal force from the ice.* **A2:** *The normal force coming from the ice and the gravitational force.* **A3:** *The* ***downward force from the earth*** *and the normal force from the ice.*
E: *The forces acting on the puck are the downward force of gravity and the upward normal force from the ice.*

8.2 Background and Related Works

8.2.1 Knowledge Graphs and its Use

Knowledge graphs have been used in several NLP tasks such as text categorization, information extraction, information retrieval, question answering and computing semantic relatedness. A knowledge graph is a relation graph consisting of edges, representing different relations and nodes, representing different entities participating in relations. An example of animal kingdom knowledge graph is shown in Figure 8.1. The figure shows a simple example with entities participating in a single relation, "BelongsToPhylum", however it should be noted that the entities could participate in multiple relations. These relations are represented by a tensor, with which various mathematical transformation could be possible. A tensor is a generalized matrix with multiple dimensions. A typical example of a 3-D tensor is shown in Figure 8.2 where each frontal slice represents one of m relations between entities E_1 through E_n.

Though knowledge graph could be used in various NLP problems such as question answering or finding semantic link between entities, automatically extracting knowledge from unstructured data is challenging. There have been a number studies such as (Angeli, Premkumar, & Manning, 2015; Jiang & Zhai, 2007; Chan & Roth, 2010), to extract knowledge from the text. These study basically

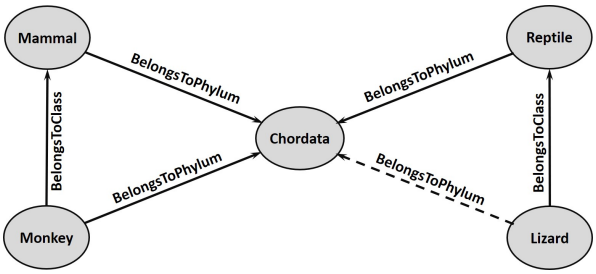

Fig. 8.1: Graphical representation of relations between entities

follow the classification approaches to classify whether the entity participate in particular relation. Implementing the methods proposed by Angelie and colleagues (Angeli et al., 2015), OpenIE, an information extraction library has been released by Stanford NLP Group[1]. This library provides various method to extract named-entity, and the information as a triplet of entities and the relations in which they participate.

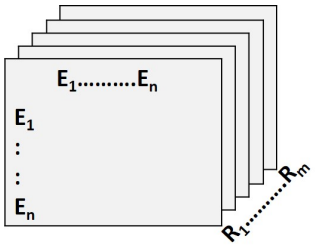

Fig. 8.2: Tensor representation of relations between entities

While knowledge graphs are directly constructed from the entity relations, one of the tasks of relational learning model is to compute the lower ranked matrix factors of the tensor to discover latent relation between entities. Similar to matrix factorization in Latent Semantic Analysis (LSA), the tensor factorization encodes the latent relations between the entities in all the relations in the tensor.

A model for relational learning is proposed by Nickel et al. (Nickel, Tresp, & Kriegel, 2011) where they presented a tensor model for relational data. They

[1]https://nlp.stanford.edu/

proposed RESCAL approach that used tensor factorization to factorize the tensor obtained from relational data to encode latent relations in the factors. This approach is comparable to LSA with two dimensional matrix representation relation between entities. However in RESCAL, representation of relations with three dimensional matrix enable to model multiple relationships between entities. For example, in case of knowledge graph of animal kingdom (Figure 8.1), it could be inferred that there exists a link between "Lizard" and "Chordata" in "BelongsToPhylum" relation. This information could be revealed by the similar connection strength of "Monkey" and "Lizard" with "Chordata" entity as they belongs to same phylum "chordata". In other words, the information from other relation ("BelongsToClass") has flowed to "BelongsToPhylum" which led to reveal the latent relationship between "Monkey" and "Lizard".

Often the knowledge bases constructed with collection of entity and relation between the entities are important resources for various tasks such as question answering and information retrieval. However a rich knowledge base also suffers from missing links(i.e. the relations) between the entities and lack of reasoning capability. Several attempts have been done to complete the knowledge bases.

Socher and colleagues (Socher, Chen, Manning, & Ng, 2013a) proposed a neural network approach to represent the relation with neural network. They developed a method to represent the entities as vectors and relations as neural tensor networks(NTN), a variant of neural network which harnesses the feed forward model with a bi-linear tensor product. The parameters of such NTN encode the latent relationship between the entities. One of the import aspect of NTN that attracted our attention towards using the model in answer assessment is that it learns the entity embedding for each of the concepts as a vector that inherently encodes the relationship with the other entities. Such embeddings of concepts could

help to discover the concepts that are correct but mentioned indirectly when compared to the reference answer in answer assessment problem.

Kotnis and Nastase (Kotnis & Nastase, 2017) showed that augmenting the relation with entity type information improves the prediction of missing links in the entity relation graph. They particularly used entity type information as type regularizer in their learning model and tested on knowledge completion task by predicting either source or target entity in Freebase knowledge graph dataset.

8.2.2 Related Works

Automated answer assessment systems have been developed in past decades. Earlier those systems (Dikli, 2006; Leacock & Chodorow, 2003; Shermis & Burstein, 2003) were mostly developed for open ended texts such as essay, where the ideas about some topics were expressed as a few paragraphs of text. On the other hand, in tutoring systems, the short answers in the form of dialogues are more common. And those short answers are are more focused and have a fixed expected answer. Unlike the essay grading, which focus more on style, coherence organization of concepts, the short answer assessment systems focus on the concept that the learners' responses carry. We mainly discuss on short answer assessment methods.

One of the earlier assessment system was OLAE, proposed by Martin and VanLehn (Martin & VanLehn, 1995). OLAE produces a student model with a collection of correct and incorrect rules from the domain model used by a particular student and Bayesian network is used to compute probable answer to a problem in the same way as the student does. One of the limitations to this approach is that the human must generate Bayesian network for each problem, i.e. the approach has a problem of scalability.

Pérez et al. (Pérez, Gliozzo, et al., 2005) applied a combination of BLEU (Papineni et al., 2002), an evaluation algorithm for machine translation, and LSA (Foltz, Laham, & Landauer, 1999) to grade student generated free text short

answers. Their main idea was to combine semantic information captured by LSA with the syntactic information captured by BLEU-inspired algorithm. They did shallow syntactic analysis such as tokenizing, parts of speech tagging and stemming. These shallow syntactic analysis though kept their system simple, portable, language independent and domain knowledge independent, the system was able to produce significantly correlated assessment score to the teacher's score.

Another work using LSA for short answer grading was by Graesser et al. (Graesser et al., 2000) in their computer literacy tutoring system: AutoTutor. They studied the quality of student contributions in tutorial dialog, and found that LSA provided evaluation that was as good as the evaluations provided by intermediate experts of computer literacy. They also found that though the LSA did not provide evaluation as high as experts in computer science, it was capable of discriminating students quality as good, vague, erroneous and mute students.

Mohler an Mihalcea (Mohler & Mihalcea, 2009) studied unsupervised text similarity techniques for short answer grading. In their work, they compared a number of knowledge-based (WordNet) and corpus-based(LSA and ESA) measures used in grading short answers to the intermediate computer science assignment questions. They studied the effect of domain and the size on corpus-based methods and found that the similarity score correlates high with human ratings when the medium sized domain-specific corpus built from Wikipedia was used. In another work, Mohler and colleague (Mohler et al., 2011) studied graph alignment methods and lexical semantic similarity methods for short answer assessment. They found that the combination of lexical semantic similarity measures and graph alignment features can grade student answer more accurately than the two methods when used individually.

Rus and Graesser (Rus & Graesser, 2006) presented lexico-syntactic information and synonymy embedded in a thesaurus to assess student answers in

AutoTutor (Graesser, Lu, et al., 2004), an Intelligent Tutoring System(ITS). They used approach of recognizing textual entailment (RTE) (Dagan, Glickman, & Magnini, 2005) by treating Expected answer as entailing text(T) and the student answer as hypothesis (H).

In other work, Rus et al. (Rus, McCarthy, Graesser, Lintean, & McNamara, 2007) studied the the performance of textual entailment (TE) method, LSA method, word overlap method and synonymy approach to assess student generated self explanation, a comprehension that student creates for a text excerpt extracted from science text. Their study showed that the TE method provided highest accuracy compared to other methods.

A major limitation of textual entailment (TE) methods proposed by Rus and colleague is that these can assess student answer as either correct or incorrect. However, in many ITS such as DeepTutor, the student answer are assessed in more than two level of correcteness. Several other studies (Mohler et al., 2011; Sultan, Salazar, & Sumner, 2016; Gomaa & Fahmy, 2012; Banjade, Maharjan, et al., 2016; Maharjan, Banjade, & Rus, 2017; Maharjan, Gautam, & Rus, 2018) were performed to assess student answers in multiple levels.

While analyzing tutorial dialogues in a dialogue based tutorial system , Niraula and colleague (Niraula et al., 2014) found that a significant portion of student answers contain pronoun that refer entities in the previous utterances. In order to address such dependency on previous context, several methods for instance (Bailey & Meurers, 2008; Banjade, Maharjan, et al., 2016; Maharjan et al., 2018, 2017) were studied before, where they assume that the question and the problem description provide import contextual cues for elliptic answers.

Bailey and Meurers (Bailey & Meurers, 2008) used previous questions as context to resolve pronoun and eliminate concepts that are given in the question. In other work, Banjade et al. developed DT-Grade dataset (Banjade, Maharjan, et al.,

2016) for assessing student answer in DeepTutor. They also presented word weighting approach, which instead of ignoring question, gives less weights to the words present in question and compute the similarity score between student answer and the reference answer. They then applied logistic regression for classifying the student answers to predict the correctness label.

Other attempt to assess student answer in DeepTutor is by Maharjan and colleague (Maharjan et al., 2017, 2018). In (Maharjan et al., 2017), they presented Gaussian Mixture Model (GMM) that used number of context aware features based on counts and word weighted lexical and alignment similarity scores. While in (Maharjan et al., 2018), they presented LSTM-based approach by capturing long-term dependencies between student answer and the previous context from problem description and the question. Both of their approach improved better than previous attempt by Banjade et al. (Banjade, Maharjan, et al., 2016), however LSTM model was better in terms of accuracy and features the model used, as no feature engineering is needed.

8.3 Methods

Our assessment system is based on a multi-class classifier that classifies a student answer into one of the four assessment labels: (i) correct, (ii) correct but incomplete, (iii) incorrect and (iv) contradictory. For this we convert each of the student answers and reference answers into feature vectors which come directly from the entities' vectors embedding learned while training Neural Tensor Network (NTN) (Socher et al., 2013a). The idea here is that once the NTN is trained, these entity vectors encode relationships among each other in the knowledge graph and the more an entity share neighbors and the relation with other entity, more similar their vector representation be. For instance, in Figure 8.3, the entity "gravitational force" and "downward force from the earth" are more likely to have similar vector embedding since they share same neighbors (i and *problem 1*) and relation

types(*constituent of*, *has head* and *is expected concept*). In the next section we will discuss methods of entity extraction, knowledge graph construction and learning entity and graph embeddings in detail. Finally we discuss the answer assment model in detail.

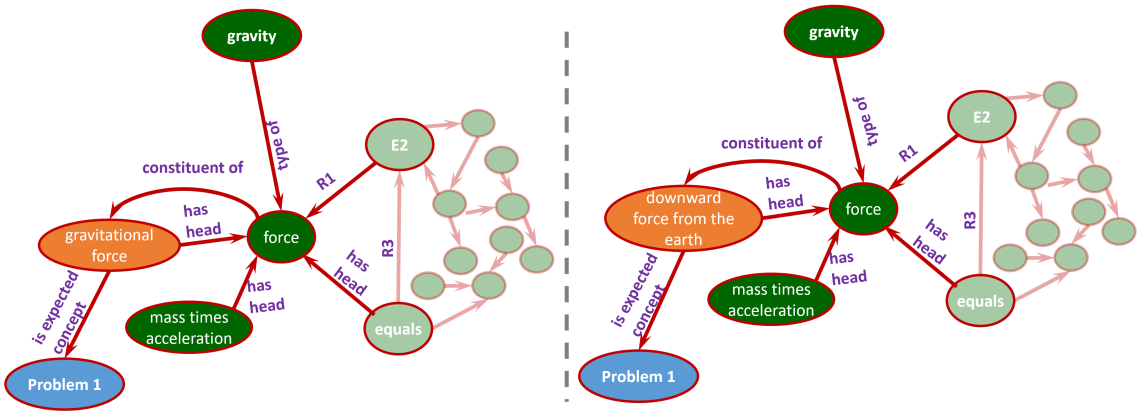

Fig. 8.3: Excerpt of knowledge graph for answer assessment

8.3.1 Entity Relations Extraction

Knowledge graph is a huge network of entities and their relationships. In order to construct the knowledge graph, a large collection of entities and relations triplets are needed. These triplets could then be used to learn latent relationships and discover missing links between the entities. In our work, we make use of two categories of such entity relations: (i) General entity relations obtained from WordNet and (ii) Domain entities relations defined and extracted from our domain dataset, i.e. Dt-Grade dataset.

The general entity relations triplets were obtained from the WordNet dataset used by Socher and colleague in their work (Socher et al., 2013a). It should be noted that the an entity represented by same word or token can have different senses. However in our work, we do learn a representation as a single entity for a token with multiple senses by merging such multi-sense tokens into a single entity.

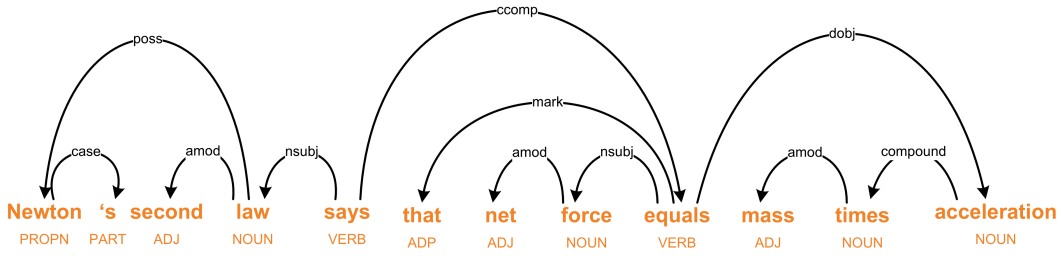

Fig. 8.4: Example of a sentence parsed with SpaCy dependency parser

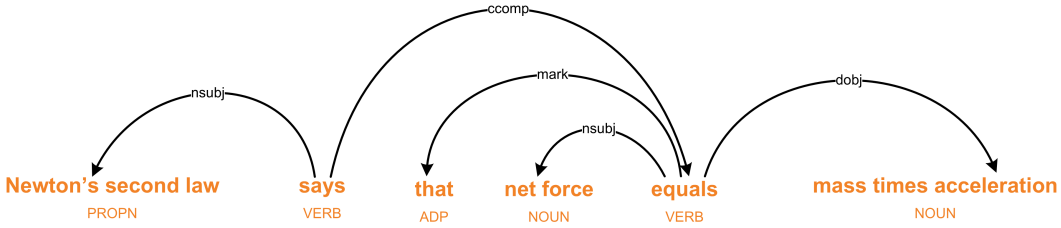

Fig. 8.5: Example of a sentence parsed with SpaCy dependency parser (phrase merged)

In order to obtain domain entity relation triplets, we assume that there are finite number of problems that are authored for the tutoring system. We then define entities and their relations for the system.

An entity is a token, a text chunk, or an unique identification number of the problem. The token entities are obtained by tokenizing the reference answers. From those tokens, we keep only content words such as nouns, verbs, adverbs and adjectives as entities. The text chunks are obtained from dependency parse tree parsed. We used SpaCy (Honnibal & Montani, 2017) for text parsing. Besides that, other kind of entities are represented by binary relationships in answers text, that are extracted using Ollie (Mausam, Schmitz, Bart, Soderland, & Etzioni, 2012), a state-of-art tool for information extraction.

In addition to extracting phrases, the dependency parse tree provides a way to define the syntactic relation between those entities. For instance, from Figures 8.4 and 8.5, we can infer several possible relations between the tokens. We define following five relations and present such relations in Table 8.2.

1. **is concept of**: relates if an entity is an expected concept of a problem. A

Table 8.2: Entities and relations extracted from example sentence *"Newton's second law says that net force equals mass times acceleration."*

Method	Entity	Relation Triplet
SpaCy	Newton second law	Newton second law : has head text : say
	net force	net force: has head text : equal
	mass time acceleration	mass time acceleration : has head text : equal
	mass	mass : is part of : mass time acceleration
	law	law : has child text : newton
	law	law : has child text : second
	Newton	Newton: has ancestor text : law
	second	second : has ancestor text : law
OLLIE	Newton second law; say; that net force equal mass time acceleration	-

problem is an abstract entity that represents problm's unique identification number ("*Problem 1*" is an abstract entity; Figure 8.3).

2. **is constituent of**: relates if an entity is constituent of another entity; i.e. if an entity token is a part of another entity text chunk ("*force*" is constituent of "*gravitational force*"; Figure 8.3)

3. **has head text**: relates if an entity's head text is another entity according to dependency parse tree.

4. **has ancestor text**: relates if an entity's ancestor is another entity according to dependency parse tree.

5. **has child text**: relates if an entity's child is another entity according to dependency parse tree.

8.3.2 Knowledge Graph Embedding

While an entity relation triplet represents a direct relation, a collection of such triplet forms a knowledge graph that provides information on indirect relation between two entities. However such knowledge graph suffers from incompleteness in the form of missing links. We discussed several approaches for discovering such

Table 8.3: Example of training triplet obtained after corrupting entities (Score = 0 for corrupted triplet)

Relation Triplet	Relation Score
(stress, type of, force)	1
(stress, type of, shell)	0
(atlantic, has part, iceland)	1
(atlantic, has part, corner	0
(atlantic, has part, north sea)	1

missing links in previous sections. In our work, we used Neural Tensor Network (NTN) proposed by Socher and colleagues (Socher et al., 2013a), which learns to identify if two entities have some kind of relationships. The way NTN relates two input entities with bilinear tensor product makes it different from the a feed forward network. The NTN architecture consists of a bilinear tensor layer as well as feed forward layer, which according to the paper, makes NTN powerful by harnessing the power of both bilinear and feed forward networks. Here we present a high level architecture (Figure 8.6) of a typical Neural Tensor Network and the scoring function (see equation 8.1) that is originally used in the original paper by Socher et al. Several NTN units (in fact equal to the number of relation types) trained in unison produces a knowledge graph embedding. Since the errors from each unit (i.e error for each relation type) are aggregated while training, the weights of each cell affect each other during training. In other words, the whole knowledge graph represented by neural tensor network gets updated. While after training, the weights of these NTN embed the relation between entities, the connection strength of two entities in the knowledge graph is given by the score function shown in equation 8.1.

$$g(e_1, R, e_2) = U_R^T f\left(e_1^T W_R^{[1:k]} e_2 + V_R \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + b_R\right) \tag{8.1}$$

where $e_1, e_2 \in \mathbb{R}^d$ are d dimensional vectors of entities, $f = tanh$, is a non linear activation function, $W_R^{[1:k]} \in \mathbb{R}^{d \times d \times k}$ is a tensor and the bilinear tensor

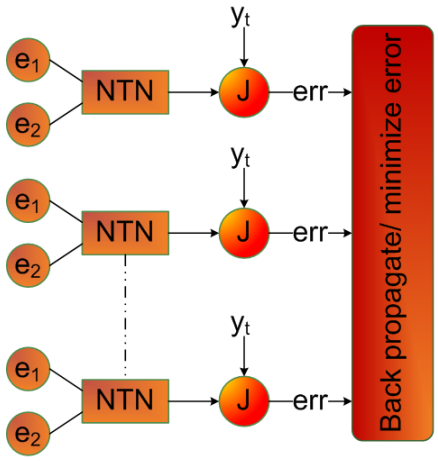

Fig. 8.6: High level architecture of knowledge graph embedding network using NTN. N NTN units for N relation types are trained in unison by minimizing the error

product $e_1^T W_R^{[1:k]} e_2$ results in a vector $h \in \mathbb{R}^k$, where each entry is computed by one slice $i = 1, ..., k$ of tensor: $h_i = e_1^T W_R^i e_2$. The other parameters for relation R are the standard form of neural network: $V_R \in \mathbb{R}^{k \times 2d}$ and $U \in \mathbb{R}^k, b_R \in \mathbb{R}^k$

To train such NTN, the negative examples are created by corrupting one of the entities in each of the positive relation triplets (see Table 8.3). Then such negative and positive triplets with corresponding binary labels are used to train the NTN. While training, the network updates its weights as well as the entity vector to obtain better representation of each of the entity after each epoch. Such vectors that are produced as bi-product are useful in our answer assessment system.

8.3.3 Classifier Using Entity Embedding

Upon completion of training the NTN, the entities vectors encode information about the relationships in the knowledge graphs. As those relations come directly from WordNet, that organize words in semantic hierarchy, and from the syntactic structure of actual domain data, it makes sense to expect that the vector captures general semantics as well as the domain specific relationship between the entities. Therefore using vectors that integrate domain general as well as domain specific information would provide better assessment model.

Using entity embedding as a basic unit, we construct vectors by extracting

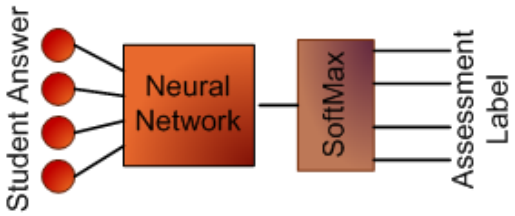

Fig. 8.7: Classifier that takes only student answer as input

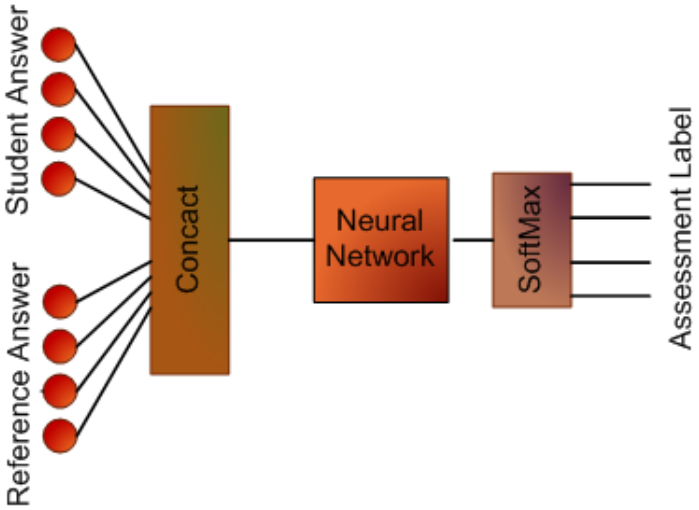

Fig. 8.8: Classifier that takes student answer as well as reference answer as input

entities from an answer instance and average the entity vectors to get a single vector for the instance. We obtain such vectors for both student answer as well as the reference answers. While applying vector average of constituent entities, the problem of out of vocabulary entity in student answer which should be handled. We address this problem by trying to replace with the synonym of the entity or its constituent tokens that are in the entity vocabulary. If none exists, we simply replace the constituent token with "*NULL*" word vector.

Our assessment model is a classifier that classifies the student answer into one of the classes that represent the assessment labels. In this work we used two types of classifiers that are based on neural network. First type is a simple neural network with one input(Figure 8.7), which takes an input vector from student

answer and produce classification. The other type has an additional layer after input layer to concatenate reference answer vector and student answer vector (Figure 8.8). The advantage of classifier with two input vectors is its ability to learn by comparing student answer with standard reference answer during training. In other words, such classifier learns to distinguish between a good answer that satisfies all the expectation with poor and bad answer that does not satisfy any of the expectations when compared to reference answer. Additionally, the reference answers are generally self contained and complete, hence they can provide contextual cues to the student answer, when used together.

Compared to one input classifier, training and predicting with two input classifier is different when there are multiple possible reference answer for same problem. For training, those reference answers, paired with corresponding student answer produce large number of training examples, a clear advantage over one input classifier. However, while predicting, multiple pairs with same true label but different predicted label could be possible for a single instance of problem (student answer). In such situation, a majority vote strategy is used to select the predicted assessment label; i.e. the assessment label that is predicted most frequently for a student answer is selected as predicted label.

8.4 Experiments and results

We performed experiments on two different types of classifiers using entity vectors learned by Neural Tensor Network trained with both domain general and domain specific relation triplets. The two types of classifiers: one input, and two inputs, trained with different entity vectors obtained from various triplet source is shown in Table 8.4. The domain general triplets are obtained from WordNet relations (prefixed with "WN"), the domain specific triplets are obtained from DT-Grade dataset (prefixed with "DT"). We also performed experiment by augmenting the domain general triplets with domain specific triplets (prefixed with

Table 8.4: Experiment models

Classifier Type	Acronym	Triplet Source
One Input	WN1IP	WordNet
	DT1IP	DT-Grade
	Aug1IP	Augmented (WordNet & DT-Grade combined)
Two Input	WN2IP	WordNet
	DT2IP	DT-Grade
	Aug2IP	Augmented (Wordnet & DT-Grade combined)

"Aug"). For augmentation, we combined the domain general entities and relation obtained from WordNet with entities and triplets obtained from DT-Grade dataset. In the following sections we first describe datasets and then present the results obtained in various experimental setups.

8.4.1 Dataset

Tutorial Dataset

We used DT-Grade dataset (Banjade, Maharjan, et al., 2016) which are extracted from logged tutorial interaction 40 junior level college students and DeepTutor system while solving conceptual physics problems. The dataset consists of 900 student responses for the same number of tutorial questions. Moreover each of the tutorial questions have a number of (approx. 5) expected answers. The student responses were labeled with the following four assessment labels:

1. Correct: Answer that covers all the expected concepts
2. Correct but incomplete: Answer that covers some of the expected concepts
3. Contradictory: Answer that is semantically opposite or contrast to expected answer.
4. Incorrect: Answer that does not include any of the expected concepts.

Though the original DT-Grade dataset consists additional information (see

Table 8.5: DT-Grade dataset

Labels	Distribution
Correct	367 (40.77%)
Incomplete	211 (23.44%)
Contradictory	84 (9.33%)
Incorrect	238 (26.44%)
Total	900

```
<Instance ID="9">
<MetaInfo StudentID ="DTSU012" TaskID="FF_LV02_PR02.sh"  DataSource="DeepTutorSummer2014"/>
<ProblemDescription>
A basketball player is dribbling a basketball
(continuously bouncing the ball off the ground).
</ProblemDescription>
<Question>
Because the ball's velocity is upward while the ball is moving upward and
its acceleration is downward, what is happening to the ball's velocity?
</Question>
<Answer>
The ball's velocity is decreasing.
</Answer>
<Annotation Label="correct(1)|correct_but_incomplete(0)|contradictory(0)|incorrect(0)">
<AdditionalAnnotation ContextRequired="0" ExtraInfoInAnswer="0"/>
<Comments Watch="0">  </Comments>
</Annotation>
<ReferenceAnswers>
1:  The ball is slowing down at a constant rate.
2:  The ball's velocity is decreasing.
3:  The ball is slowing down.
</ReferenceAnswers>
```

Fig. 8.9: An annotation example of DT-Grade dataset

Figure 8.9); we only use student answers, reference answers and the labels for student answers. Table 8.5 shows the distribution of labels in the dataset.

Knowledge Graph Dataset

We use wordnet knowledge graph dataset that was used by Socher and colleagues (Socher, Chen, Manning, & Ng, 2013b). We preprocess the wordnet triplets to combine the different sense for same word into a single entity for training our neural tensor network. Though the different senses are combined, the relation that those different sense words previously participated was kept unchanged and treated as a separate training instance. This makes the model simple yet encoding the relations in the embedding. There are 11 relations categories obtained from WordNet. These categories characterize the semantic relations between the entities in knowledge graph. Besides that we created entity relation triplets dataset from reference answers in DT-Grade dataset. The entities we created are of two types: (i)

Table 8.6: Knowledge graph dataset. The number of triplets are the actual number of positive instances (doubles after corrupting the triplets)

source	#Entity	#Triplet
WN	33163	109165
DT	1263	22941
Combined	34352	132106

the question itself is the entity; i.e. there are 900 such entities. (ii) The content words, phrases, head words, parents and children obtained by parsing the reference answers using SpaCy (Honnibal & Montani, 2017) dependency parser. After obtaining the entities, we identified 5 syntactic relations among the entities obtained from reference answers (see Table 8.7). We used these two categories of knowledge graph dataset separately as well as we augment the syntactic knowledge graph dataset by combining with semantic knowledge graph dataset. Table 8.6 shows the number of entities and relations in different knowledge graph dataset we used.

Table 8.7: Semantic and surface relations

Type	Relations
Semantic	has instance
	type of
	member meronym
	member holonym
	part of
	has part
	subordinate instance of
	domain region
	synset domain topic
	similar to
	domain topic
Surface	is concept of
	is constituent of
	has head text
	has ancestor text
	has child text

8.4.2 Results

The results of 10-folds cross validation is summarized in Table 8.8. We report the performance in terms of accuracy, and F1 measure. The result shows that Aug2IP, performed best with average accuracy of 0.644, which is 2.2% better

than *LSTM, our previous best performing model(0.622) (Maharjan et al., 2018). Also its F1 score (=0.642) is 2.2%, and Kappa (=0.482) is 3.2% better than of *LSTM. The previous model (*LSTM) used LSTM that take problem description, tutor question, student answer and reference answer as input, however, this work learns entity or word vectors to discover general semantic and domain specific linguistic relationships. In fact, our two input classifier in this work when used with domain specific vectors (DT2IP & Aug2IP) performed better. This suggests that the NTN model could learn vectors better than the one-hot-encoding used in previous approach.

Besides performing better than previous model, the result suggests that when trained with vectors created from same dataset, the classifiers that takes both student answer answer and reference answer as input perform better compared to that only takes student answer as input. For instance, DT2IP has average accuracy of 0.626 which is 5.7% higher than of DT1IP. Similarly Aug2IP has average accuracy of 0.644 which is 4% higher than Aug1IP. Whereas the performance of WN2IP is higher than of WN1IP, it is small improvement (1.3%) when compared to other classifiers.

Table 8.8 further shows that the classifier when trained domain specific vectors (prefixed with DT) perform better than domain general vectors prefixed with WN). Moreover, when the domain specific triplets were augmented with domain general triplets, the performance boosted up significantly (3.5% improvement for Aug1IP than DT1IP, and 1.8% for Aug2IP than DT2IP).

Figure 8.10 shows the average precision, recall and F1 score of various model we experimented. As seen from the figure, our two input classifier trained with augmented vector performed best in terms of precision(0.639), recall(0.644) and F1 score. Compared to domain specific vectors (DT1IP and DT2IP) the domain general vectors(WN1IP and WN2IP) performed worse. The reason could be because

significant number of the entities extracted from student answers and reference answers from DT-Grade dataset were not present in domain general vocabulary. And lack of such entities resulted inaccurate representation of entities. Because such entities need either semantically similar entities or synonym words and even further may have to rely on NONE entity in the vocabulary.

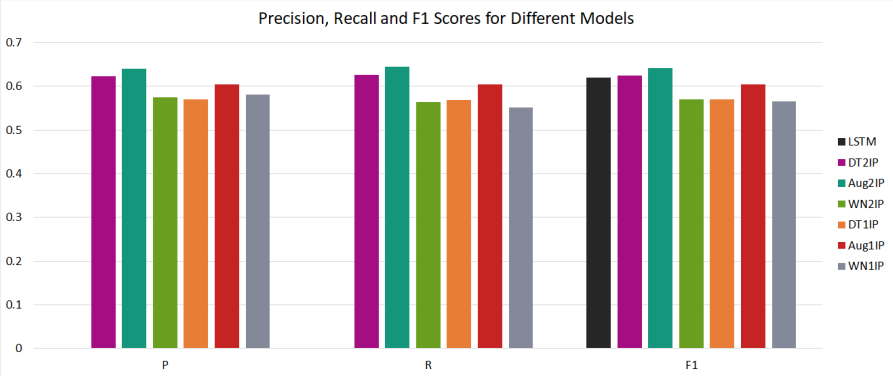

Fig. 8.10: Comparison of precision, recall and f1 score for different models

Table 8.8: Performance of models

Model	Avg acc	F1	Kappa
*LSTM (Maharjan et al., 2018)	0.622	0.620	0.450
DT2IP	**0.626**	**0.624**	0.450
Aug2IP	**0.644**	**0.642**	0.482
WN2IP	0.564	0.569	0.334
DT1IP	0.569	0.569	0.350
Aug1IP	0.604	0.604	0.409
WN1IP	0.551	0.565	0.302

8.5 Conclusions

In this work we proposed different knowledge graph based models to assess student responses in DeepTutor. The improved performance in terms of accuracy and F1 of our models suggests knowledge graph yields better vectorial representation of student answer and reference answer text. In addition, the two input classifier always performed better than one input classifier when trained with same set of vectors. This is expected, since the two input classifier uses reference

answer as one input which provides context for student answers. More importantly when the two input classifier is trained with the augmented vectors, they performed best. This suggest that that the relation triplets obtained from actual tutorial data helps to encode highly predictive features while training NTN.

Our method has several areas where further improvement is possible. One of those area is to define more relations in Newtonian physics domain. In this work we have limited to the syntactic analysis of the text to discover relations. In future we plan to study on automatically discovering more relations from free text that are more semantic than syntactic.

Chapter 9

Conclusion and Future Works

9.1 Conclusions

Our dissertation is motivated towards reducing gaps between ITS and CPS researches by adopting the techniques that are successful in ITS fields. To achieve this, we have studied various aspects of interactions between learners and the learning systems, and proposed different assessment models to assess textual contents generated by learners. We particularly concentrated our analysis in two categories of systems, one the virtual internships where learners are focused on gaining skills in virtual companies and the responses are more open ended with limited guidance from the teacher. Other systems' interaction we analyzed is physics tutoring system DeepTutor, where students are guided by computer tutor in every step of learning.

For automatically assessing notebooks in engineering virtual internships, we developed different models by using mainly for categories of features namely essay scoring features, domain expert features, LIWC, and Co-Metrix. All models performed very well with good and very good kappa scores (kappas scores of 0.6-0.8 are considered very good). Our results show that, in this context, the predictive value of models using only the general text analysis features is comparable to the predictive value of a model using only the DE features. In particular, the ES group of features is the best predictor of students' justifications quality. When other groups of features are added to the individual ES model, the results do not improve significantly. The fact that the ES features are so good is not surprising. Word count, or essay length, which is one of the features in the ES group, is known as being the best predictor of essay quality in automated essay grading. Also, the

CohMetrix group of features are a good predictor of the quality of students' justifications.

It is important to note, however, that the predictive power of a model is only one dimension for evaluating the utility of automated assessment models in learning environments like virtual internships. We suggest that developmental cost and interpretability of the models are also valuable dimensions to consider. Of the models presented above, those using only the general text analysis features have the lowest developmental cost. Moreover, these features are generally applicable across types of tasks, specific tasks, and domains. In contrast, models containing the DE features are more specific to engineering virtual internships, have a relatively high developmental cost because their features required the time and expertise of humans to develop.

Besides those methods to assess open ended responses using general text analysis and domain expert features, we also developed model for more constrained(requiring criteria) text assessment. For this, we investigated a method for creating classifiers for virtual internship notebook entries using teacher provided specifications without the use of participant data. These classifiers used LSA based and NN based semantic similarity methods to capture the general semantic relationships among concepts. We also investigated regular expression based classifiers. The results are impressive in the sense that some classifiers, using both LSA and NN, gave high precision and recall values using thresholds derived without participant data, which suggests that model development using small number of teacher created specification is plausible.

Moreover, for the classifiers that use LSA based semantic similarity methods, we also analyzed the impact of corpus size, corpus specificity, and semantic space dimensionality on the performance of the assessment methods. Our analysis showed that the LSA spaces generated from a domain specific corpus can perform better

when compared to a space generated from the much larger TASA corpus for the notebook assessment task. For the domain specific corpus, the best average performance over all the target concepts was obtained for the maximum available corpus size and the maximum number of dimensions. However, this performance is comparable to results obtained with a smaller corpus size and smaller vector dimensionality indicating that smaller corpora and spaces can be good enough to boost assessment components for virtual internships.

Additionally, to understand conversation pattern and the students' mastery of epistemic frames, we conducted a Markov process analysis of students' conversation. We have experimented and validated our method on data from an engineering epistemic frame using eight different ways to model Markov processes for each student participating in engineering virtual internships. Our analysis showed that most of the students attain similar SKIVE profile after spending sufficiently long time in virtual internships. The comparison of the distribution of SKIVE elements for individual students in models with noState revealed that some students may play more managerial or coordinator roles than others.

To further understand the conversation pattern in virtual internships, we explored several methods for speech act classification. We explored various classifier models with different categories of features as well as training strategies. We found that the latent features generated by a pre-trained sentence embeddings model (derived from a large Wikipedia corpus) yielded better performance compared to the other models. Besides that, the predictive power of the neural network model was further boosted when pre-trained with noisy label before training with expert-annotated data. In summary, representing chat utterances with vector embeddings could be a better candidate method for analyzing conversation in the system where there is no clear turn taking between the speakers and the analysis

has to rely mostly on the content of the utterance itself rather than the previous chat utterances.

We also explored various LSTM models and configurations for handling negation in tutorial dialogues. We experimented and validated our models using real dialogues between student and an intelligent tutoring system. Our experiments suggests that the sequence to sequence tagger can handle well the substasks of negation scope, focus and cue detection, outperforming a previous method based on CRF that relied on human engineered features. A good choice of hyper-parameters of LSTM such as dropouts, number of units and layers could result in a competitive model for negation handling.

While studying various aspects of interactions in learning system, this dissertation has made substantial contributions towards developing intelligent Collaborative Problem Solving system that will deliver effective, personalized, and cost-effective instruction to all learners at any time with access to an internet-connected device.

9.2 Future Works

For further advancement of the research in assessment of learners response and automating the mentoring and tutoring process in learning systems, we plan to work on one or more of the follwing areas.

While assessing notebooks in virtual internships, we are considering unsupervised methods to automatically detect domain specific codes that could be used as features in our DE models. Furthermore, we are considering unsupervised topic detection in student-generated justification as a way to generalize the applicability of our models to other domains and types of tasks.

We will investigate a method to combine the classifiers that only use teacher created notebooks for assessing student notebooks, in order to better understand

how performance of one model is boosted by another in the scenario where participants responses vary widely compared to the sample responses.

We plan to build LSTM based negation handling models that could be trained with both past and future contexts. We also plan to experiment with a hybrid model that uses word embeddings as well as human engineered features.

We plan to use the stationary distribution of SKIVE components obtained from the analysis we have done so far to better understand students' effectiveness of acquiring much needed skills to be successful professionally. Furthermore, we plan to develop a dual Markov process to infer stationary distributions of states and transitions.

We plan to expand the speech act classification model we studied by using more contextual information. Given the multi-party nature of our conversation data, before we can use contextual information, it is necessary to disentangle the conversations into sets of related utterances. Our future models will disentagle the multi-party conversations before attempting to use contextual information for speech act classification.

References

Angeli, G., Premkumar, M. J. J., & Manning, C. D. (2015). Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)* (Vol. 1, pp. 344–354).

Arastoopour, G., Chesler, N. C., D'Angelo, C. M., Shaffer, D. W., Opgenorth, J. W., Reardan, C. B., ... Lepak, C. G. (2012). Nephrotex: Measuring first-year students' ways of professional engineering thinking in a virtual internship. In *American society for engineering education.*

Austin, J. L. (1975). *How to do things with words.* Oxford university press.

Bagley, E., & Shaffer, D. W. (2009). When people get in the way: Promoting civic thinking through epistemic gameplay. *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)*, *1*(1), 36–52.

Bailey, S., & Meurers, D. (2008). Diagnosing meaning errors in short answers to reading comprehension questions. In *Proceedings of the third workshop on innovative use of nlp for building educational applications* (pp. 107–115).

Banjade, R., Maharjan, N., Niraula, N. B., Gautam, D., Samei, B., & Rus, V. (2016). Evaluation dataset (dt-grade) and word weighting approach towards constructed short answers assessment in tutorial dialogue context. In *Proceedings of the 11th workshop on innovative use of nlp for building educational applications* (pp. 182–187).

Banjade, R., Niraula, N. B., & Rus, V. (2016). Towards detecting intra-and inter-sentential negation scope and focus in dialogue. In *Flairs conference* (pp. 198–203).

Bellegarda, J. R. (2005). Latent semantic mapping [information retrieval]. *IEEE signal processing magazine*, *22*(5), 70–80.

Blanco, E., & Moldovan, D. (2011). Semantic representation of negation using focus detection. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1* (pp. 581–589).

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Bodin, M. (2012). Mapping university students' epistemic framing of computational physics using network analysis. *Physical Review Special Topics-Physics Education Research*, *8*(1), 010115.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Bradford, R. B. (2008). An empirical study of required dimensionality for large-scale latent semantic indexing applications. In *Proceedings of the 17th acm conference on information and knowledge management* (pp. 153–162).

Cai, Z., Graesser, A., Forsyth, C., Burkett, C., Millis, K., Wallace, P., ... Butler, H. (2011). Trialog in aries: User input assessment in an intelligent tutoring system. In *Proceedings of the 3rd ieee international conference on intelligent computing and intelligent systems* (pp. 429–433).

Cai, Z., Graesser, A., Windsor, L., Cheng, Q., Shaffer, D. W., & Hu, X. (2018). Impact of corpus size and dimensionality of LSA spaces from wikipedia articles on autotutor answer evaluation. In *Proceedings of the 11th international conference on educational data mining, EDM 2018, buffalo, ny, usa, july 15-18, 2018*. Retrieved from `http://educationaldatamining.org/files/conferences/EDM2018/papers/EDM2018\_paper\_89.pdf`

Chan, Y. S., & Roth, D. (2010). Exploiting background knowledge for relation extraction. In *Proceedings of the 23rd international conference on computational linguistics* (pp. 152–160).

Chesler, N. C., Bagley, E., Breckenfeld, E., West, D., & Shaffer, D. W. (2010). A virtual hemodialyzer design project for first-year engineers: An epistemic game approach. In *Asme 2010 summer bioengineering conference* (pp. 585–586).

Chesler, N. C., Ruis, A., Collier, W., Swiecki, Z., Arastoopour, G., & Shaffer, D. W. (2015). A novel paradigm for engineering education: Virtual internships with individualized mentoring and assessment of engineering thinking. *Journal of biomechanical engineering*, *137*(2), 024701.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

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).

Councill, I. G., McDonald, R., & Velikovich, L. (2010). What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the workshop on negation and speculation in natural language processing* (pp. 51–59).

Crossley, S. A., Dascalu, M., & McNamara, D. S. (2017). How important is size? an investigation of corpus size and meaning in both latent semantic analysis and latent dirichlet allocation. In *30th international florida artificial intelligence research society conference, flairs 2017.*

Dagan, I., Glickman, O., & Magnini, B. (2005). The pascal recognising textual entailment challenge. In *Machine learning challenges workshop* (pp. 177–190).

D'Angelo, C., Arastoopour, G., Chesler, N., Shaffer, D. W., et al. (2011). Collaborating in a virtual engineering internship. In *Computer supported collaborative learning conference, hong kong sar, hong kong, china, july* (pp.

4–8).

Davies, M. (2010). The corpus of contemporary american english as the first reliable monitor corpus of english. *Literary and linguistic computing*, *25*(4), 447–464.

Dikli, S. (2006). An overview of automated scoring of essays. *The Journal of Technology, Learning and Assessment*, *5*(1).

Dredze, M., Wallach, H. M., Puller, D., & Pereira, F. (2008). Generating summary keywords for emails using topics. In *Proceedings of the 13th international conference on intelligent user interfaces* (pp. 199–206). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/1378773.1378800`

Ezen-Can, A., & Boyer, K. E. (2013). Unsupervised classification of student dialogue acts with query-likelihood clustering. In *Educational data mining 2013.*

Fernando, S., & Stevenson, M. (2008). A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th annual research colloquium of the uk special interest group for computational linguistics* (pp. 45–52).

Foltz, P. W., Laham, D., & Landauer, T. K. (1999). The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, *1*(2), 939–944.

Forsyth, E. N. (2007). *Improving automated lexical and discourse analysis of online chat dialog* (Unpublished doctoral dissertation). Monterey, California. Naval Postgraduate School.

Freedman, R., Ali, S., & McRoy, S. (2000). What is an intelligent tutoring system. *intelligence*, *11*(3), 15–16.

Gautam, D., Zachari Swiecki, D. W., Graesser, A. C., & Rus, V. (2017). Modeling classifiers for virtual internships without participant data. , 278–283.

Givón, T. (1993). *English grammar: A function-based introduction* (Vol. 2). John Benjamins Publishing.

Glaser, B., & Strauss, A. (2009). The discovery of grounded theory: Strategies for qualitative research: Transaction publishers.

Gomaa, W. H., & Fahmy, A. A. (2012). Short answer grading using string similarity and corpus-based similarity. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *3*(11).

Gong, Y., & Liu, X. (2001). Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th annual international acm sigir conference on research and development in information retrieval* (pp. 19–25).

Graesser, A. C., Hu, X., Nye, B. D., VanLehn, K., Kumar, R., Heffernan, C., ... others (2018). Electronixtutor: an intelligent tutoring system with multiple learning resources for electronics. *International Journal of STEM Education*, *5*(1), 15.

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), 180–192.

Graesser, A. C., McNamara, D. S., Louwerse, M. M., & Cai, Z. (2004). Coh-metrix: Analysis of text on cohesion and language. *Behavior research methods, instruments, & computers*, *36*(2), 193–202.

Graesser, A. C., Wiemer-Hastings, P., Wiemer-Hastings, K., Harter, D., Tutoring Research Group, T. R. G., & Person, N. (2000). Using latent semantic analysis to evaluate the contributions of students in autotutor. *Interactive learning environments*, *8*(2), 129–147.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Herrenkohl, L. R., & Cornelius, L. (2013). Investigating elementary students'

scientific and historical argumentation. *Journal of the Learning Sciences*, *22*(3), 413–461.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Honnibal, M., & Montani, I. (2017). spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Horn, L. (1989). A natural history of negation.

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd acm international conference on conference on information & knowledge management* (pp. 2333–2338).

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Huddleston, R., Pullum, G. K., et al. (2002). The cambridge grammar of english. *Language. Cambridge: Cambridge University Press*, 1–23.

Jessup, E., & Martin, J. (2001). Taking a new look at the latent semantic analysis approach to information retrieval. *Computational information retrieval*, *2001*, 121–144.

Jiang, J., & Zhai, C. (2007). A systematic exploration of the feature space for relation extraction. In *Human language technologies 2007: The conference of the north american chapter of the association for computational linguistics; proceedings of the main conference* (pp. 113–120).

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition.* Prentice Hall, Pearson Education International.

Kim, J., Chern, G., Feng, D., Shaw, E., & Hovy, E. (2006). Mining and assessing

discussions on the web through speech act analysis. In *Proceedings of the workshop on web content mining with human language technologies at the 5th international semantic web conference.*

Kim, S. N., Cavedon, L., & Baldwin, T. (2012). Classifying dialogue acts in multi-party live chats. In *Proceedings of the 26th pacific asia conference on language, information, and computation* (pp. 463–472).

Konstantinova, N., De Sousa, S. C., Díaz, N. P. C., López, M. J. M., Taboada, M., & Mitkov, R. (2012). A review corpus annotated for negation, speculation and their scope. In *Lrec* (pp. 3190–3195).

Kontostathis, A. (2007). Essential dimensions of latent semantic indexing (lsi). In *System sciences, 2007. hicss 2007. 40th annual hawaii international conference on* (pp. 73–73).

Kotnis, B., & Nastase, V. (2017). Learning knowledge graph embeddings with type regularizer. *arXiv preprint arXiv:1706.09278*.

Landauer, T. K. (2006). Latent semantic analysis. In *Encyclopedia of cognitive science.* American Cancer Society. Retrieved from `https://onlinelibrary.wiley.com/doi/abs/10.1002/0470018860.s00561`

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, *25*(2-3), 259–284.

Landauer, T. K., Laham, D., Rehder, B., & Schreiner, M. E. (1997). How well can passage meaning be derived without using word order? a comparison of latent semantic analysis and humans. In *Proceedings of the 19th annual meeting of the cognitive science society* (pp. 412–417).

Leacock, C., & Chodorow, M. (2003). C-rater: Automated scoring of short-answer questions. *Computers and the Humanities*, *37*(4), 389–405.

Lidstone, G. J. (1920). Note on the general case of the bayes-laplace formula for

inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, *8*(182-192), 13.

Lintean, M. C., & Rus, V. (2012). Measuring semantic similarity in short texts through greedy pairing and word semantics. In *Flairs conference.*

Maharjan, N., Banjade, R., & Rus, V. (2017). Automated assessment of open-ended student answers in tutorial dialogues using gaussian mixture models. In *The thirtieth international flairs conference.*

Maharjan, N., Gautam, D., & Rus, V. (2018). Assessing free student answers in tutorial dialogues using lstm models. In *International conference on artificial intelligence in education* (pp. 193–198).

Martin, J., & VanLehn, K. (1995). Student assessment using bayesian nets. *International Journal of Human-Computer Studies*, *42*(6), 575–591.

Mausam, Schmitz, M., Bart, R., Soderland, S., & Etzioni, O. (2012). Open language learning for information extraction. In *Proceedings of conference on empirical methods in natural language processing and computational natural language learning (emnlp-conll).*

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mohler, M., Bunescu, R., & Mihalcea, R. (2011). Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1* (pp. 752–762).

Mohler, M., & Mihalcea, R. (2009). Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the 12th conference of the european chapter of the association for computational linguistics* (pp. 567–575).

Moldovan, C., Rus, V., & Graesser, A. C. (2011). Automated speech act

classification for online chat. *MAICS*, *710*, 23–29.

Morante, R., & Blanco, E. (2012). * sem 2012 shared task: Resolving the scope and focus of negation. In *Proceedings of the first joint conference on lexical and computational semantics-volume 1: Proceedings of the main conference and the shared task, and volume 2: Proceedings of the sixth international workshop on semantic evaluation* (pp. 265–274).

Morante, R., Schrauwen, S., & Daelemans, W. (2011). Annotation of negation cues and their scope: Guidelines v1. *Computational linguistics and psycholinguistics technical report series, CTRS-003*.

Mutalik, P. G., Deshpande, A., & Nadkarni, P. M. (2001). Use of general-purpose negation detection to augment concept indexing of medical documents: a quantitative study using the umls. *Journal of the American Medical Informatics Association*, *8*(6), 598–609.

Nickel, M., Tresp, V., & Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Icml* (Vol. 11, pp. 809–816).

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 *Lrec* (pp. 3199–3203).

Nkambou, R., Mizoguchi, R., & Bourdeau, J. (2010). *Advances in intelligent tutoring systems* (Vol. 308). Springer Science & Business Media.

Nwana, H. S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, *4*(4), 251–277.

Olney, A., Louwerse, M., Matthews, E., Marineau, J., Hite-Mitchell, H., & Graesser, A. (2003). Utterance classification in autotutor. In *Proceedings of the hlt-naacl 03 workshop on building educational applications using natural language processing-volume 2* (pp. 1–8).

Pagliardini, M., Gupta, P., & Jaggi, M. (2017). Unsupervised learning of sentence

embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345–1359.

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).

Pennebaker, J. W., Francis, M. E., & Booth, R. J. (2001). Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, *71*(2001), 2001.

Penumatsa, P., Ventura, M., Graesser, A. C., Louwerse, M., Hu, X., Cai, Z., & Franceschetti, D. R. (2006). The right threshold value: What is the right threshold of cosine measure when using latent semantic analysis for evaluating student answers? *International Journal on Artificial Intelligence Tools*, *15*(05), 767–777.

Pérez, D., Alfonseca, E., Rodríguez, P., Gliozzo, A., Strapparava, C., & Magnini, B. (2005). About the effects of combining latent semantic analysis with natural language processing techniques for free-text assessment. *Revista signos*, *38*(59), 325–343.

Pérez, D., Gliozzo, A. M., Strapparava, C., Alfonseca, E., Rodriguez, P., & Magnini, B. (2005). Automatic assessment of students' free-text answers underpinned by the combination of a bleu-inspired algorithm and latent semantic analysis. In *Flairs conference* (pp. 358–363).

Pröllochs, N., Feuerriegel, S., & Neumann, D. (2016). Negation scope detection in sentiment analysis: Decision support for news-driven trading. *Decision Support Systems*, *88*, 67–75.

Rokach, L., Romano, R., & Maimon, O. (2008). Negation recognition in medical

narrative reports. *Information Retrieval*, *11*(6), 499–538.

Rupp, A. A., Choi, Y., Gushta, M., Mislevy, R., Bagley, E., Nash, P., ... Shaffer, D. (2009). Modeling learning progressions in epistemic games with epistemic network analysis: Principles for data analysis and generation. In *Proceedings from the learning progressions in science conference* (pp. 24–26).

Rus, V., D'Mello, S., Hu, X., & Graesser, A. (2013). Recent advances in conversational intelligent tutoring systems. *AI magazine*, *34*(3), 42–54.

Rus, V., Feng, S., Brandon, R. D., Crossley, S. A., & McNamara, D. S. (2011). A linguistic analysis of student-generated paraphrases. In *Flairs conference.*

Rus, V., Gautam, D., Swiecki, Z., Shaffer, D. W., & Graesser, A. (2016). Assessing student-generated design justifications in virtual engineering internships. In *Edm* (pp. 496–501).

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., Lintean, M., & Azevedo, R. (2009). Automatic detection of student mental models during prior knowledge activation in metatutor. *International working group on educational data mining*.

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: System demonstrations* (pp. 163–168).

Rus, V., Lintean, M. C., Graesser, A. C., & McNamara, D. S. (2009). Assessing student paraphrases using lexical semantics and word weighting. In *Aied* (pp. 165–172).

Rus, V., Maharjan, N., Tamang, L. J., Yudelson, M., Berman, S., Fancsali, S. E., & Ritter, S. (n.d.). An analysis of human tutors' actions in tutorial dialogues.

Rus, V., McCarthy, P. M., Graesser, A. C., Lintean, M. C., & McNamara, D. S. (2007). Assessing student self-explanations in an intelligent tutoring system. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 29).

Rus, V., Moldovan, C., Niraula, N., & Graesser, A. C. (2012). Automated discovery of speech act categories in educational games. *International Educational Data Mining Society*.

Rus, V., & Niraula, N. (2012). Automated detection of local coherence in short argumentative essays based on centering theory. In *International conference on intelligent text processing and computational linguistics* (pp. 450–461).

Rus, V., Stefanescu, D., Niraula, N., & Graesser, A. C. (2014). Deeptutor: towards macro-and micro-adaptive conversational intelligent tutoring at scale. In *Proceedings of the first acm conference on learning@ scale conference* (pp. 209–210).

Salton, G., & Lesk, M. E. (1968). Computer evaluation of indexing and text processing. *Journal of the ACM (JACM)*, *15*(1), 8–36.

Samei, B., Li, H., Keshtkar, F., Rus, V., & Graesser, A. C. (2014). Context-based speech act classification in intelligent tutoring systems. In *International conference on intelligent tutoring systems* (pp. 236–241).

Searle, J. R. (1969). *Speech acts: An essay in the philosophy of language* (Vol. 626). Cambridge university press.

Shaffer, D., Borden, F., Srinivasan, A., Saucerman, J., Arastoopour, G., Collier, W., ... Frank, K. (2015). The ncoder: A technique for improving the utility of inter-rater reliability statistics. *Games and Professional Simulations Technical Report*, *1*.

Shaffer, D. W. (2006a). Epistemic frames for epistemic games. *Computers & education*, *46*(3), 223–234.

Shaffer, D. W. (2006b). *How computer games help children learn*. Macmillan.

Shaffer, D. W., Hatfield, D., Svarovsky, G. N., Nash, P., Nulty, A., Bagley, E., ... Mislevy, R. (2009). Epistemic network analysis: A prototype for 21st-century assessment of learning. *International Journal of Learning and Media*, *1*(2).

Shaffer, D. W., Ruis, A., & Graesser, A. C. (2015). Authoring networked learner models in complex domains. In *Design recommendations for intelligent tutoring systems: authoring tools* (pp. 179–191). US Army Research Laboratory, Orlando (FL).

Shen, Y., He, X., Gao, J., Deng, L., & Mesnil, G. (2014). A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management* (pp. 101–110).

Shermis, M. D., & Burstein, J. C. (2003). *Automated essay scoring: A cross-disciplinary perspective*. Routledge.

Socher, R., Chen, D., Manning, C. D., & Ng, A. (2013a). Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems* (pp. 926–934).

Socher, R., Chen, D., Manning, C. D., & Ng, A. Y. (2013b). Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems 26*.

Stefănescu, D., Banjade, R., & Rus, V. (2014). Latent semantic analysis models on wikipedia and tasa. In *Language resources evaluation conference (lrec)*.

Steinberger, J., & Ježek, K. (2004). Using latent semantic analysis in text summarization and summary evaluation. In *In proc. isim '04* (pp. 93–100).

Sultan, M. A., Salazar, C., & Sumner, T. (2016). Fast and easy short answer grading with high accuracy. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: Human language technologies* (pp. 1070–1075).

Sutskever, I., Martens, J., & Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th international conference on machine learning (icml-11)* (pp. 1017–1024).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104–3112).

Swiecki, Z. M., & Shaffer, D. (2017). Dependency-centered design as an approach to pedagogical authoring. *Game-based learning: Theory, strategies and performance outcomes*.

Tierney, L. (1994). Markov chains for exploring posterior distributions. *the Annals of Statistics*, 1701–1728.

Tisi, J., Whitehouse, G., Maughan, S., & Burdett, N. (2013). A review of literature on marking reliability research (report for ofqual). *Slough: NFER*.

Tottie, G. (1993). Negation in english speech and writing: A study in variation. *Language*, *69*(3), 590–593.

Vanlehn, K. (2006). The behavior of tutoring systems. *International journal of artificial intelligence in education*, *16*(3), 227–265.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist*, *46*(4), 197–221.

Vincze, V., Szarvas, G., Farkas, R., Móra, G., & Csirik, J. (2008). The bioscope corpus: biomedical texts annotated for uncertainty, negation and their scopes. *BMC bioinformatics*, *9*(11), S9.

Wang, H. C., & Chiu, Y. F. (2011). Assessing e-learning 2.0 system success. *Computers & Education*, *57*(2), 1790–1800.

Wedin, M. V. (1990). Negation and quantification in aristotle. *History and Philosophy of Logic*, *11*(2), 131–150.

Widdows, D., & Peters, S. (2003). Word vectors and quantum logic: Experiments with negation and disjunction. *Mathematics of language*, *8*(141-154).

Williams, C., & D'Mello, S. (2010). Predicting student knowledge level from domain-independent function and content words. In *International conference on intelligent tutoring systems* (pp. 62–71).

Yeh, J.-Y., Ke, H.-R., Yang, W.-P., & Meng, I.-H. (2005). Text summarization using a trainable summarizer and latent semantic analysis. *Information processing & management*, *41*(1), 75–95.

Zhu, M., & Zhang, M. (2016). Examining the patterns of communication and connections among engineering professional skills in group discussion: A network analysis approach. In *Integrated stem education conference (isec), 2016 ieee* (pp. 181–188).