Advancement Auto-Assessment of Students Knowledge States from Natural Language Input

Nisrine Ait Khayi

Follow this and additional works at: https://digitalcommons.memphis.edu/etd

Recommended Citation
https://digitalcommons.memphis.edu/etd/2397

This Dissertation is brought to you for free and open access by University of Memphis Digital Commons. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of University of Memphis Digital Commons. For more information, please contact khggerty@memphis.edu.
Advancement Auto-Assessment of Students` Knowledge States from Natural Language Input

A Dissertation

Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

Major: Computer Science

The University of Memphis

November 5th, 2021
Dedication

To my beloved sister Aoiatif
ACKNOWLEDGMENTS

I would like to thank dearly my dissertation’s advisor and chair Dr. Vasile Rus for his unlimited academic, financial, and personal support during my PhD studies. Without this, this dissertation work would not be accomplished. I have enjoyed immensely my graduate experience.

I would like to also thank my dear committee members: Dr. Fleming, Dr. Venugopal and Dr. Zhang. Their fruitful feedback and discussions were vital in improving the outcome of this dissertation research.

I would like to thank the University of Memphis and Institute for Intelligent Systems for their generous financial support to attend conferences which was beneficial in disseminating my work and getting feedback and new research ideas to successfully complete my dissertation work.

I would like also to thank my lab colleagues for their collaboration and cooperation during my PhD studies. Their feedback and technical help were significant for my research’s success.

Finally, I thank dearly my beloved mom and sisters for their unconditional love and support in this academic journey. The countless daily long videocalls were helpful to provide me with the needed encouragement and support in this journey. I hope they will be proud of this achievement as the first daughter to get a PhD.
Knowledge Assessment is a key element in adaptive instructional systems and in particular in Intelligent Tutoring Systems because fully adaptive tutoring presupposes accurate assessment. However, this is a challenging research problem as numerous factors affect students’ knowledge state estimation such as the difficulty level of the problem, time spent in solving the problem, etc. In this research work, we tackle this research problem from three perspectives: assessing the prior knowledge of students, assessing the natural language short and long students’ responses, and knowledge tracing.

Prior knowledge assessment is an important component of knowledge assessment as it facilitates the adaptation of the instruction from the very beginning, i.e., when the student starts interacting with the (computer) tutor. Grouping students into groups with similar mental models and patterns of prior level of knowledge allows the system to select the right level of scaffolding for each group of students. While not adapting instruction to each individual learner, the advantage of adapting to groups of students based on a limited number of prior knowledge levels has the advantage of decreasing the authoring costs of the tutoring system. To achieve this goal of identifying or clustering students based on their prior knowledge, we have employed effective clustering algorithms.

Automatically assessing open-ended student responses is another challenging aspect of knowledge assessment in ITSs. In dialogue-based ITSs, the main interaction between the learner and the system is natural language dialogue in which students freely respond to various system prompts or initiate dialogue moves in mixed-initiative dialogue systems. Assessing freely generated student
responses in such contexts is challenging as students can express the same idea in different ways owing to different individual style preferences and varied individual cognitive abilities. To address this challenging task, we have proposed several novel deep learning models as they are capable to capture rich high-level semantic features of text.

Knowledge tracing (KT) is an important type of knowledge assessment which consists of tracking students’ mastery of knowledge over time and predicting their future performances. Despite the state-of-the-art results of deep learning in this task, it has many limitations. For instance, most of the proposed methods ignore pertinent information (e.g., Prior knowledge) that can enhance the knowledge tracing capability and performance. Working toward this objective, we have proposed a generic deep learning framework that accounts for the engagement level of students, the difficulty of questions and the semantics of the questions and uses a novel times series model called Temporal Convolutional Network for future performance prediction.

The advanced auto-assessment methods presented in this dissertation should enable better ways to estimate learner’s knowledge states and in turn the adaptive scaffolding those systems can provide which in turn should lead to more effective tutoring and better learning gains for students. Furthermore, the proposed method should enable more scalable development and deployment of ITSs across topics and domains for the benefit of all learners of all ages and backgrounds.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td><strong>Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>Intelligent Tutoring Systems</td>
<td>1</td>
</tr>
<tr>
<td>Deep Learning</td>
<td>4</td>
</tr>
<tr>
<td>Research Challenges</td>
<td>6</td>
</tr>
<tr>
<td>Research Questions and Scope</td>
<td>10</td>
</tr>
<tr>
<td><strong>Clustering Students based on their Prior Knowledge</strong></td>
<td>14</td>
</tr>
<tr>
<td>Introduction</td>
<td>14</td>
</tr>
<tr>
<td>Related Work</td>
<td>15</td>
</tr>
<tr>
<td>Background</td>
<td>17</td>
</tr>
<tr>
<td>Clustering Methods</td>
<td>18</td>
</tr>
<tr>
<td>DP-means Algorithm</td>
<td>18</td>
</tr>
<tr>
<td>K-Modes Clustering</td>
<td>20</td>
</tr>
<tr>
<td>Experiments</td>
<td>22</td>
</tr>
<tr>
<td>Dataset</td>
<td>22</td>
</tr>
<tr>
<td>Experiments: Binary Data</td>
<td>24</td>
</tr>
<tr>
<td>Experiments: Clustering Categorical Data</td>
<td>30</td>
</tr>
<tr>
<td>Conclusions</td>
<td>32</td>
</tr>
<tr>
<td><strong>Assessment of Open-Ended Student Answers Using Bi-GRU Capsule Networks</strong></td>
<td>34</td>
</tr>
<tr>
<td>Introduction</td>
<td>34</td>
</tr>
<tr>
<td>Related Prior Work</td>
<td>38</td>
</tr>
<tr>
<td>Model Architecture</td>
<td>40</td>
</tr>
<tr>
<td>Embedding Layer</td>
<td>40</td>
</tr>
<tr>
<td>Bi-GRU Layer</td>
<td>42</td>
</tr>
<tr>
<td>Capsule Layer</td>
<td>43</td>
</tr>
<tr>
<td>Experiments</td>
<td>45</td>
</tr>
<tr>
<td>DT-Grade Dataset</td>
<td>45</td>
</tr>
<tr>
<td>Experimental Setting</td>
<td>47</td>
</tr>
<tr>
<td>Hyperparameters</td>
<td>48</td>
</tr>
<tr>
<td>Results &amp; Discussion</td>
<td>48</td>
</tr>
<tr>
<td>Conclusions</td>
<td>49</td>
</tr>
<tr>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td><strong>An-Attention Based Transformer Model for Student Answers Assessment</strong></td>
<td>51</td>
</tr>
<tr>
<td>Introduction</td>
<td>51</td>
</tr>
<tr>
<td>Related Prior Work</td>
<td>53</td>
</tr>
<tr>
<td>Model Architecture</td>
<td>56</td>
</tr>
<tr>
<td>Embedding Layer</td>
<td>56</td>
</tr>
<tr>
<td>Positional Encoding</td>
<td>57</td>
</tr>
<tr>
<td>Transformer Layer</td>
<td>58</td>
</tr>
<tr>
<td>Multi-Head Self-Attention Mechanism</td>
<td>59</td>
</tr>
<tr>
<td>Position-Wise Feed-Forward Networks</td>
<td>62</td>
</tr>
<tr>
<td>Experiments</td>
<td>62</td>
</tr>
<tr>
<td>Data</td>
<td>63</td>
</tr>
<tr>
<td>Experimental Setting</td>
<td>64</td>
</tr>
<tr>
<td>Hyperparameters</td>
<td>65</td>
</tr>
<tr>
<td>Results &amp; Discussion</td>
<td>65</td>
</tr>
<tr>
<td>Conclusions</td>
<td>69</td>
</tr>
</tbody>
</table>

**Towards Assessing Open-Ended Students Answers Using Graph Convolutional Networks** | 71   |
| Introduction                                                        | 71   |
| Related Work                                                        | 72   |
| Model Architecture                                                  | 74   |
| DT-Grade Graph                                                      | 74   |
| Graph Convolutional Networks (GCN)                                  | 76   |
| SoftMax Layer                                                       | 77   |
| Experiments                                                         | 78   |
| Experimental Setting                                                | 78   |
| Results & Discussion                                                | 79   |
| Conclusions                                                         | 81   |

**Towards Improving Open Student Answer Assessment using Pretrained Transformers** 82
| Introduction                                                        | 82   |
| Related Work                                                        | 83   |
| Methods                                                             | 84   |
| Experiments and Results                                             | 85   |
| Conclusions                                                         | 87   |

**Discourse based Automated Essay Scoring using XLNET Model** 88
| Introduction                                                        | 88   |
| Related Work                                                        | 90   |
| Proposed Model                                                      | 92   |
| Experiments and Results                                             | 96   |
| ASAP dataset                                                        | 96   |
| Experimental Settings                                              | 97   |
| Evaluation Metric                                                   | 98   |
| Results and analysis                                                | 98   |
| Conclusions                                                         | 100  |
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Examples of student generated short answers during tutorial dialogues</td>
<td>7</td>
</tr>
<tr>
<td>2. Categorial Data</td>
<td>23</td>
</tr>
<tr>
<td>3. Binary Data</td>
<td>23</td>
</tr>
<tr>
<td>4. Clustering results of DP-means with different types of distances</td>
<td>26</td>
</tr>
<tr>
<td>5. DP-means clustering results with different values of $\alpha$</td>
<td>26</td>
</tr>
<tr>
<td>6. K-means results</td>
<td>28</td>
</tr>
<tr>
<td>7. Agglomerative clustering results</td>
<td>28</td>
</tr>
<tr>
<td>8. DP-means clustering results (categorical data) with different values of $\alpha$</td>
<td>31</td>
</tr>
<tr>
<td>9. K-modes results with $k = 3$</td>
<td>31</td>
</tr>
<tr>
<td>10. Annotation example of the DT-Grade dataset</td>
<td>46</td>
</tr>
<tr>
<td>11. The distribution of classes in training (800 instances) and testing data (100 instances)</td>
<td>47</td>
</tr>
<tr>
<td>12. The performance results of various models</td>
<td>49</td>
</tr>
<tr>
<td>13. Feed Forward Network Architecture</td>
<td>62</td>
</tr>
<tr>
<td>14. Results of variations of the Transformer architecture using word2vec embeddings.</td>
<td>66</td>
</tr>
<tr>
<td>15. Results of variations of the Transformer architecture using Glove embeddings.</td>
<td>66</td>
</tr>
<tr>
<td>16. Results of variations of the Transformer architecture using Elmo embeddings.</td>
<td>67</td>
</tr>
<tr>
<td>17. Results of variations of the Transformer architecture using FastText embeddings.</td>
<td>67</td>
</tr>
<tr>
<td>18. Comparison between the transformer and other deep learning models.</td>
<td>68</td>
</tr>
<tr>
<td>19. Example of students and reference answers and their assessment using Bi-GRU and ELMo-Transformer</td>
<td>69</td>
</tr>
</tbody>
</table>
20. Performance of GCN with different filters 80
21. Performance of GCN with different number of units 80
22. Comparison between GCN and other models for the DT-Grade dataset 81
23. Performance results of the pretrained models 87
24. Statistics of the ASAP dataset; Range means the score range 98
25. Experimental Results 100
26. Experimental results without Coh-Metrix features 101
27. Cognitive Tutor Datasets Statistics 114
28. The performance results of the proposed model and the ablation experiments results 116
29. Example of the results in terms of engagement level and difficulty of the question 116
30. Comparison of performance results between our proposed model and the existing DKT methods using the Cognitive Tutor datasets 117
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. DeepTutor Framework is based on the following loops: i)- Outer Loop corresponds to task loop, ii)- Inner Loop refers to step solution loop and iii)- Hint Loop</td>
<td>3</td>
</tr>
<tr>
<td>2. DP-means algorithm</td>
<td>20</td>
</tr>
<tr>
<td>3. K-modes Clustering Algorithm</td>
<td>22</td>
</tr>
<tr>
<td>4. Quality of the DP-means algorithm using different values of $\alpha$</td>
<td>26</td>
</tr>
<tr>
<td>5. DP-means visualization with $\alpha = 3$</td>
<td>27</td>
</tr>
<tr>
<td>6. DP-means visualization with $\alpha = 3.1$</td>
<td>27</td>
</tr>
<tr>
<td>7. DP-means visualization with $\alpha = 3.2$</td>
<td>28</td>
</tr>
<tr>
<td>8. Quality of DP-means algorithm using different values of $\alpha$</td>
<td>29</td>
</tr>
<tr>
<td>9. Students’ answers assessment problem statement</td>
<td>37</td>
</tr>
<tr>
<td>10. Bi-GRU-Capsnet model architecture: it consists of the following layers (i) embedding layer, (ii)- Bi-GRU layer, (iii)- Capsule Layer and (iv) SoftMax Layer</td>
<td>42</td>
</tr>
<tr>
<td>11. Capsule structure</td>
<td>43</td>
</tr>
<tr>
<td>12. The Routing Algorithm</td>
<td>44</td>
</tr>
<tr>
<td>13. The attention mechanism allows to encode the word “it” based on focusing on “The animal” words.</td>
<td>52</td>
</tr>
<tr>
<td>14. The attention-based Transformer architecture. It consists of the following components: (i)-an embedding layer, (ii) a Transformer Encoder, and (iii) a SoftMax layer.</td>
<td>57</td>
</tr>
<tr>
<td>15. Position Encoding Example</td>
<td>58</td>
</tr>
<tr>
<td>16. Transformer architecture that consists of several identical encoders (left). Encoder structure (right).</td>
<td>59</td>
</tr>
<tr>
<td>17. Scaled dot product attention (left). Multi-Head Attention Structure (right)</td>
<td>60</td>
</tr>
<tr>
<td>18. Snapshot of the DT-Grade raw dataset in XML form. The main are attributes: (i) Problem Description, (ii)-Question and(iii) Reference Answers</td>
<td>63</td>
</tr>
</tbody>
</table>
19. The model architecture consists of the following components: i)- building a DTGraph, ii)- feeding it to two GCN layers, iii)- and finally applying a classifier.

20. Model Architecture

21. Model Architecture

22. Engagement level evolution of a student at different time steps while interacting with Cognitive Tutor.

23. The architecture of the Universal Sentence Encoder

24. Evolution of knowledge states of a student over time

25. Mastery level of two students in six concepts
Chapter 1

Introduction

Intelligent Tutoring Systems

In 1984, Bloom (1984) has conducted a study demonstrated that students that studied under the guidance of a human tutor combined with traditional instructions have performed two standard deviations (sigma) better than those who received traditional group teaching. Motivated by this result, the intelligent tutoring systems (ITS) have emerged (Nkambou, Bourdeau, & Mizoguchi, 2010) as computer-based systems that promote more adaptive and individualized approach for instruction (Martin, 1999; Ahuja, & Sille, 2013). Over several years of research, the ITSs have been deployed successfully to optimize the learning gains and enhance learning in numerous domains such as Science, Technology, Engineering and Maths (STEM) in a personalized and adaptive environment (AbuEl-Reesh, 2018; Agha et al., 2018; Al-Nakhal et al., 2017; Al Rekhawi et al., 2018; Leelawong et al., 2008; Qwaider et al., 2018).

Based on the literature, the architecture of an ITS is based on the following key components:

(i) – a domain model that represents all the knowledge that the designer intended to be learned by students,

(ii) - a student model that reflects the most current state of knowledge,

(iii) - a tutor model that prepares the suitable content for each student. In each instructional moment, it addresses the specific “changing cognitive needs of the individual learner” and intervenes in students’ activities, when necessary (Ohlsson, 1986, p. 293), and

(iv) - a user interface that serves as a communication interface between the system and the student. Several research studies showed that the most effective tools for learning within dialog-based systems is providing a personalized curriculum and personalized feedback as they simulate a familiar learning environment of student–tutor interaction, thus helping to improve student motivation.
(Al-Dahdooh et al., 2017; Al-Nakhal et al., 2017; Albacete et al., 2019; Chi et al., 2011; Munshi et al., 2019; Rus et al., 2014a; Rus et al., 2014b).

In this work, our focus is the state of art of intelligent tutoring systems DeepTutor (Rus et al., 2013, Rus et al., 2015) which is a dialog-based system that promotes deep learning of complex science topics through a combination of advanced domain modeling methods, deep language and discourse processing algorithms, and advanced tutorial strategies. DeepTutor is based on constructivist theories of learning and Socratic principles of instruction. That is giving students the opportunity to self-explain the solutions and giving the system the opportunity to correct misconceptions through hints and appropriate feedback (positive feedback – “Great job.”; negative feedback – “This is incorrect.”; neutral feedback - “Ok.” etc.). DeepTutor is full adaptive dialog system (Rus et al., 2014c). It offers micro-adaptivity that refers to a system’s capability to adapt its scaffolding while the learner is working on a particular task. It also offers macro-adaptivity that refers to a system’s capability to select appropriate instructional tasks for the learner to work on based on his mastery level. The development of the full adaptability of DeepTutor is guided and explained by its instructional framework as described in Figure 1. Following the ITS architecture proposed by Vanlehn (2006), Deep Tutor behavior can be described by the following three loops:

- **Task loop**: selects the next task to work on.
- **Solution-step loop**: manages the student-system interaction while the student works on a particular task. It loops through the steps in a solution and initiates a hint loop for each missing step in a student solution.
• **Hint loop**: enacts strategies that help students construct missing steps in the solution by themselves with minimal help from the system based on constructivist theories of learning.

Figure 1. DeepTutor Framework is based on the following loops: i)- Outer Loop corresponds to task loop, ii)- Inner Loop refers to step solution loop and iii)- Hint Loop

DeepTutor relies on several important components such as: i) a dialogue-based task script, ii)- learner models, iii) - models for handling dialogue acts and dialogue modes, iv) assessment models for assessing the student answer, and v) a diagnostic feedback model. In this work, we focus on the knowledge assessment module.
Deep Learning

Machine learning methods have drawn a lot of attention in recent years. Most of these classical based methods rely heavily on the process of features engineering, that is extracting the most representative features for the algorithms to work, discarding noninformative attributes. Some of the popular hand-crafted features include bag of words (BoW) model that represents text as a bag of its words, discarding grammar and word order and keeping word frequencies. These classical machines learning based methods suffer from several limitations. For instance, depending on the hand-crafted features requires a tedious and time-consuming feature engineering process to obtain a good performance (Minaee et al., 2020). In addition, designing the features depends on the knowledge domain that makes these methods difficult to easily generalize to new tasks. Finally, these models can’t take full advantage of the large training datasets because the features are pre-defined.

Deep learning breaks away all the above difficulties using deep and layered model structure, often in the form of neural networks and the associated end-to-end learning algorithms. Deep learning enables the model to learn from patterns and examples (Nawaz et al., 2012, Wang et al., 2011) and offers feature learning to automatically discover the representations needed for the task at hand (Zhong et al., 2016).

Deep learning has made impressive advances in various fields such as computer vision. Following this trend, several NLP researchers applied extensively deep learning to obtain the state-of-the-art results in various tasks such as text classification, questions answering etc. (Clark et al., 2020; Raffel et al., 2019; Lan et al., 2019; Sanh et al., 2019, Liu et al., 2019). As deep learning networks can’t process text, these models consist of an embedding layer. For instance, word embeddings are set of language modeling and feature learning techniques in NLP where
words or phrases from the vocabulary are mapped to vectors of real numbers. Typically, words with similar meaning will have close vector representation in the embedding space. More importantly, word embedding’s goal is capturing a sort of relationship such as meaning, context and morphology.

Based on their diverse architectures, deep learning models can be classified into several categories such as:

- **Recurrent Neural Networks (RNN):** consider text as sequence of words. It is effective in capturing text dependencies and structures.
- **Convolutional Neural Networks (CNN):** use convolutions and pooling operations to capture patterns in text.
- **Capsule Networks:** overcome the pooling problem of CNN networks in losing information about the local order of words.
- **Attention Networks:** The attention mechanism has been effective in capturing correlated words in a text.
- **Graph Neural Networks:** have been effective in capturing the internal graph structures of natural language, such as syntactic and semantic parse trees.
- **Transformers based models:** boosted the performance of many NLP tasks via the paradigm of pretraining-fine tuning.
- **Hybrid based models:** combine attention and other approaches (e.g., CNN, RNN) and have been utilized to capture effectively local and global semantic features of text (e.g. order of words).

Despite the criticism of the deep learning approach of being a black box that lacks interpretability, and its requirement of large computational resources, we explore its potential for
the students answers assessment and knowledge tracing tasks as it has demonstrated a superior performance over traditional machine learning methods.

**Research Challenges**

Assessment is a key element in education in general and in Intelligent Tutoring Systems (ITSs; Rus et al. 2013) because fully adaptive tutoring presupposes accurate assessment (Chi et al. 2001; Woolf 2008). The quality of information about the student’s knowledge toward a target topic facilities the personalization of learning in an ITS (Brusilovsky, 1996). This information is acquired through the assessment process that consists of measuring what the student knows about the taught subject. However, this is a challenging research problem as numerous factors affects this knowledge estimation such as the difficulty level of the problem, time spent in solving the problem, the motivation of the student etc. In this research work, we tackle this problem from three perspectives: assessing the prior knowledge of students, assessing the natural language short and long students’ responses and knowledge tracing.

DeepTutor offers a prior knowledge assessment by the means of the pretest in form of multiple-choice questions (Rus et al.,2016). Implementing a pretest in an ITS serves two primary objectives: computing the learning gains when combined with the posttest and facilitates the macro-adaptation in the ITS through selecting the appropriate task for the learner based on his/her current knowledge state. Macro-adaptation can be expensive if the number of unique student knowledge states is very large as it requires selecting a unique set of tasks for each such unique knowledge state. Considering each of these potential knowledge states and selecting for each corresponding learner a unique set of tasks becomes a computationally and authoring challenge. An alleviation option would be to group students into clusters of similar mental models and prior
knowledge then select and author tasks for each such clusters. That is, grouping students into similar mental model groups can offer a good trade-off between adaptivity and authoring costs.

Automatically assessing open-ended short student responses is another challenging aspect of knowledge assessment in ITS. In dialogue-based systems, the main interaction between the learner and the system is natural language dialogue in which students freely responses to various system prompts or initiate dialogue moves in mixed-initiative dialogue systems. Assessing freely generated student’ responses in such contexts is challenging as students can express the same idea in different ways owing to different individual style preferences and varied individual characteristics such as cognitive abilities and knowledge. Table 1 shows four answers, articulated by four different college students, to a question asked by a state-of-the-art conversational ITS DeepTutor. It should be noted that all four student answers in Table 1 are correct answers to the tutor question. As can be seen from the table, some students write full sentences (student answer A4), some others write very short answers (A3), and yet other students write elaborate answers that include additional concepts relative to the reference answer (A1).

Table 1. Examples of student generated short answers during tutorial dialogues

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Problem description</strong>: While speeding up, a large truck pushes a small compact car. <strong>Tutor question</strong>: How do the magnitudes of forces they exert on each other compare? <strong>Reference answer</strong>: The forces from the truck and car are equal and opposite. <strong>Student answers</strong>: A1. The magnitudes of the forces are equal and opposite to each other due to Newton’s third law of motion. A2. they are equal and opposite in direction. A3. equal and opposite A4. the truck applies an equal and opposite force to the car.</td>
</tr>
</tbody>
</table>
Assessing the freely generated student answers in conversational tutoring can be achieved using various approaches. Semantic similarity is a widely adopted and scalable approach in which the student answer is compared to a reference answer produced by an expert. Typically, a normalized semantic similarity score, between 1 and 5 is computed. A high score implies the correctness of the student answer. Despite the high effectiveness of semantic similarity in several NLP tasks such as text summarization (Wang et al., 2008; Nenkova et al., 2011), question answering (Vo et al., 2015) and machine translation (Corley and Mihalcea, 2005), this approach suffers from several challenges such as the variability of natural language as mentioned previously (Banjade et al., 2016; Maharjan et al., 2018; Ait Khayi et al., 2019). In addition, it has been found that students use frequently pronouns (e.g., it, his, her etc.) in their answers to refer to concepts mentioned in the tutor question or problem description. Therefore, using a contextual information is vital to solve this reference matching problem.

Automated Essays Scoring (AES) task can be viewed as an extension of short student answers assessment to long essays assessment. Most of the research work done in this area is based on a holistic approach which summarizes the quality of an essay with a single score. A major limitation of this approach is its inability to identify what aspect of the essay needs improvement. To alleviate this limitation and provide constructive feedback to students, researchers started to assess specific traits of the quality of an essay such as organization (Persing et al., 2010; Taghipouret al., 2017; Mathias et al. 2018; Song et al. 2020), sentence clarity (Persing et al., 2013; Ke et al. 2019), prompt adherence (Persing et al., 2014), argument strength (Persing et al., 2015; Taghipour et al., 2017) etc. Although, discourse is one of the most important aspects of written essays, less attention has been paid to incorporate the discourse structure into the distributed representation of essays. The traditional methods used to capture the discourse structure of the
essay, are based on annotations, and parsing which are very time consuming. Thus, using a pretrained transformer that processes long sequences efficiently can reduce the cost. In addition, parsing texts is challenging when the texts are poorly written which is the case of the Automated Student Assessment Prize (ASAP) dataset that we are using in this work.

In general, the AES models can be divided to two main streams. The first stream are the feature engineering-based models, which are driven by handcrafted features such as the grammatical errors. The second stream are deep learning-based models that are effective in extracting deep semantic features. In this work, we argue that these two streams should be considered complementary since the neural approach cannot encode some features that are handcrafted and vice-versa. Therefore, proceeding with a hybrid approach that integrates both models can be promising in terms of performance and capturing richer discourse structure of essays.

Another challenging aspect of knowledge assessment is knowledge tracing (KT) which consists of tracking the students’ mastery of knowledge over time and predicting their future performances. This task is usually leveraged to optimize students’ learning trajectories, experiences, and outcomes. It is a challenging task due to the complexity of the human learning processes (e.g., guessing, forgetting etc.) and the inherent difficulties of modeling knowledge (e.g., prior background; (Piech et al., 2015)). Further improvements in KT, which is the focus of our work, will have a wide range of benefits including better adaptation to individual learner’s needs and, consequently, improved effectiveness at inducing learning gains and better learning experiences. Despite the state-of-the-art results obtained by various deep learning models applied in this task, they suffer from major limitations. For instance, many of the existing deep KT models do not account for other relevant information such as the number of the correct attempts to solve
a task and the duration of each step, which can be viewed as indicators of levels of engagement. Therefore, incorporating pertinent information (e.g., engagement level) into deep learning models would be beneficial in enhancing the KT capability and performance.

Our current research work consists of solving these major research challenges to facilitate the adaptation of the instruction for each individual learner and improve his overall tutoring experience within intelligent tutoring systems.

**Research Questions and Scope**

This dissertation work was driven by finding the most effective ways of assessing the student knowledge to enable the adaptivity of the instruction, improve the effectiveness of the tutoring experience within dialog-based systems and optimize the learning gains of students. Therefore, the current research work is guided by answering the following research questions.

**Research Question 1:** What are the most effective ways of assessing the knowledge of students within dialog-based systems?

Answering this research question presents our main research contribution of this dissertation work. We address this problem from three perspectives: (i) assessing the prior knowledge of students using effective clustering algorithms, (ii) assessing the open-ended students answers using novel deep learning-based models, and (iii) knowledge tracing.

**Research Question 2:** How can we achieve a tradeoff between the adaptivity and authoring costs within intelligent tutoring systems?

We introduce effective clustering algorithms in chapter 2 and in Ait Khayi et al. (2019). We propose to group students with similar prior knowledge patterns using several effective clustering algorithms such as DP-means and K-modes. This grouping allows the system to assign the same
task with a specific level of knowledge to the same group lowering the authoring costs of the ITS. In addition, understanding the gaps in specific cluster can inform the macro-adaptivity of the system.

**Research Question 3:** *Can a Capsule Network based model achieve an accurate assessment of open-ended natural language answers within DeepTutor?*

Capsule Networks have been introduced to address the problem of Convolutional Neural Networks of losing relevant information such as local order of words via the pooling operations. Its application in the NLP field, especially in text classification, has resulted in a significant performance gain. Motivated by this success, we introduce a Bi-GRU-Capsule Neural Network based model for student answer assessment task in chapter 3 and in Ait khayi et al. (2019).

**Research Question 4:** *Can an Attention-based Transformer model improve the performance results of assessing the short students answers within DeepTutor?*

The major limitation of the previous Capsule Network based model is its incapability in assessing the very short answers and the answers with pronouns referring to previous concepts. To mitigate the shortness problem, we propose in chapter 4 and in Ait Khayi et al. (2020), to use an Attention based Transformer proposed by Vaswani et al. (2017) that utilizes a multi-head attention. This mechanism enables the overall assessment based on the most relevant words of the student answer and reference answer. To alleviate the reference matching problem, we proposed injecting a context, as the concatenation of the problem description and the question, into the student answer.

**Research Question 5:** *Can a Graph Convolutional Network be utilized to achieve an accurate assessment of short students’ answers?*
Graph Neural Networks have been effective at tasks that have rich relational structure and can preserve global structure information of a graph in graph embeddings. As this can be beneficial for the student answers assessment task, we propose in chapter 5 and in (Ait Khayi et al., 2020), a Graph Convolutional Network (GCN) to assess open-ended students answers within the state of art of intelligent tutoring systems DeepTutor. In this work, we proceed as a node classification task. Following the citation approach, we constructed a knowledge graph where nodes represent physics questions, whereas the edges represent the similarities between these questions. Then, we imported this graph to two layers of GCN to generate nodes embeddings. Finally, these nodes embeddings are fed to a classification layer.

**Research Question 6:** *Can Finetuning the pretrained transformers improve the current performance results of the short students answers assessment downstream task?*

The pretraining-finetuning paradigm has revolutionized the natural language processing field yielding state-of the art results in several subfields such as text classification, question answering and semantic similarity. Motivated by these successes, we propose in chapter 6 and in (Ait khayi et al., 2021) to fine tune several pretrained transformers on the student answers assessment downstream task.

**Research Question 7:** *How to effectively assess the discourse aspect of long essays?*

Applying a hybrid approach that integrates the features engineering approach and the neural approach has advanced the performance results in the Automated Essay Scoring (AES) research area. Added to this, assessing the traits of the quality essay allows to provide constructive feedback to learners on what aspect of the essay needs improvements. Since, a little attention has paid to the discourse trait of the essays, we propose in chapter 7 a hybrid approach to assess the discourse aspect of essays. First, we extract the essay representation using the pretrained XLNET model.
Then, we concatenate it with selected discourse handcrafted features (e.g., Lexical chains). Finally, we feed the merged vector to a linear layer to predict the final score.

**Research Question 8: How to enhance the performance and the capability of the knowledge tracing task?**

Incorporating relevant information such as prior knowledge into deep learning models has advanced the performance results of the knowledge tracing task. Motivated by these successes, we propose in chapter 8 a generic framework that accounts for the engagement level (for the first time) of students, the difficulty of questions, and the questions semantics in learning the knowledge embeddings. These embeddings sequences are then passed to an LSTM based model to learn the hidden states of the knowledge of students, which are passed to a Temporal Convolutional Network to predict the future performances.
Chapter 2

Clustering Students based on their Prior Knowledge

Introduction

Capturing students’ knowledge state, our focus, and other learner characteristics that are important for learning such as their emotional state is critical to facilitate learning through adaptivity, i.e., tailoring instruction to each individual learner (Shute et al., 2012). It should be noted that adaptivity can be thought of at two levels: macro-adaptivity which means selecting appropriate instructional tasks and what are the predominant misconceptions they hold? and are these misconceptions evenly distributed across topics and level of course taught? (Also called within-task adaptivity). Our work presented here could inform both micro- and macro-adaptivity. For instance, understanding the knowledge gaps of students in a particular cluster could inform what instructional tasks to choose for these students, i.e., it informs macro-adaptivity.

Indeed, an important preliminary step in creating an ITS that is sensitive to student misconceptions and individual learning trajectories is to first understand the various levels of mastery with respect to a target domain, for instance, physics. For example, important questions that need to be answers are: What are the predominant misconceptions they hold? and are these misconceptions evenly distributed across topics and level of course taught? Using the clustering method proposed here will help answer such important questions. To this end, we document for each group of students identified by our clustering algorithm, the major misconceptions exhibited by that group.

In this study, we applied clustering on a pretest data collected at the beginning of an experiment in which high-school students interacted with a dialogue-based ITS. Our goal was to identify student groups and analyze them as a group in terms of misconceptions and mastered
concepts. The identified groups could then be used to inform the authoring of instructional tasks and within-task instructional strategies and feedback for each group as opposed to each learner, which would be a much more expensive process. Learning such individualized strategies for each learner would be possible using automated methods, such as reinforcement learning, but they require substantially more experimental data which it is not have available.

The main clustering algorithm used in this study is the DP-means algorithm (Kulis et al., 2012). Its main advantage is identifying the number of clusters using a Dirichlet Process Mixture Model. After briefly presented related work and the context of our own work, what follows is a description of the DP-means and k-modes algorithms. We then present details of the experiments and results. We conducted experiments using two types of data: binary and categorical responses. In addition, other clustering algorithms were employed to compare the results with those obtained with DP-means. We evaluated the performance of the resulted clusters using intrinsic, e.g., based on the silhouette index which measures the compactness of each cluster, and extrinsic methods, e.g., based on students’ post-test scores derived from post-test responses which were not used to generate the clusters.

**Related Work**

Clustering has been used in the past for analyzing education data as indicated by the research studies presented next. Bouchet and colleagues (2013) have applied the Expectation-Maximization clustering algorithm on data collected from the MetaTutor ITS. MetaTutor scaffolds student’s metacognitive skills while learning about the human circulatory system. The main objective of their clustering was reinforcing self-regulated learning via student profiling. The results consisted of three distinct clusters of students in terms of performance. The results have been analyzed using a MANOVA approach. Rodrigo and colleagues (2008) have applied k-means clustering on data
collected from students interacting with Aplusix, an ITS for Algebra. The main research goal of their work was identifying students’ behaviors through an analysis of interaction logs. The results have demonstrated the existence of two clusters of students associated with differing behavior and affective states. The first cluster reflected more collaborative work whereas the second cluster reflected more solitary work. Reyes-Gonzalez and colleagues (2018) have used the LC-Conceptual clustering algorithm from logical combinatorial pattern recognition for student modeling in an ITS. This algorithm is based on two phases: the first phase consists of building groups of objects based on their similarity and a grouping criterion. The second phase is called the intentional structure phase where the distinctive features of each resulted cluster are determined. Fang and colleagues (2018) have used k-means clustering to capture learning patterns in over 250 students who used AutoTutor to gain reading comprehension skills. The average response times per question and performance across lessons have been used to cluster the students’ learning behavior. The results showed the convergence of four types of learners: proficient readers, struggling readers, conscientious readers and disengaged readers. Classifying readers can improve the adaptivity of AutoTutor ITS by providing a proactive feedback and intervention based on the learning behaviors.

Similar to those other approaches, our intention was to discover groups of students with similar knowledge states as characterized by their responses to the multiple-choice pre-test. Each incorrect choice in the pre-test is associated with a major misconception and therefore students that pick similar choices should be assigned to the same cluster. The centroid of the cluster could then be used to interpret the strengths and weaknesses of students in that cluster and appropriate interventions designed for that group.
Background

Our work was conducted in the context of an experiment in which high-school students interacted with a dialogue-based intelligent tutoring system that tutors students on science topics through problem-solving. The system encourages students to self-explain solutions to complex science problems and only offers help, in the form of hints, when needed, e.g., when the student is floundering. That is, during a typical tutorial session, the system challenges students to solve a number of problems that are carefully selected by the system in order to optimize student learning (macro-adaptivity). When working on a particular problem, students are first asked to provide a solution that must include a justification based on concepts and principles of the target domain, which was Newtonian Physics in the case of our study presented here. All other things equal, low knowledge students will most likely struggle to provide solid self-explanations and most likely to articulate misconceptions which would lead to more scaffolding dialogue moves in terms of hints and correcting misconceptions on the part of the computer tutor (micro-adaptation). High knowledge students would need less scaffolding and therefore the corresponding dialogues should be shorter.

Before students start interacting with the system, they took a pre-test to assess their initial knowledge state. The tool elected to assess students’ initial knowledge state was an enhanced version of the Force Concept Inventory (FCI). The Force Concept Inventory (FCI) is a 30-item multiple-choice "test" designed to assess student understanding of the most basic concepts in Newtonian mechanics (Halloun, Hake, and Mosca, 1995). The FCI presents students with various situations and ask them to choose between Newtonian explanations for the phenomena, versus common-sense alternatives (Hestenes, Wells, & Swackhamer, 1992). The FCI has been widely used to measure learning in introductory physics courses. For example, Hake (1998) reported FCI
data from 6,000 high school and university students. Coletta and Phillips (2005) combined their data with data collected by Hake (1998) and in combination used the FCI to measure learning in 73 university and college introductory physics classes. The data we have is based on an augmented version of the FCI consisting of 35 multiple-choice questions. The augmented FCI adds a number of questions for certain Newtonian topics which were not covered enough in the original FCI test.

We administered the augmented Force Concept Inventory (aFCI) to students at three public and two private high schools in the mid-south region, including six teachers and 26 classrooms. The pretest was administered in classroom. Students completed the aFCI via provided scantron sheets, which were then collated and processed. The results of the scantron sheets were then compared to direct markings on the actual aFCI test in the case of blank or unidentifiable scantron responses. The data collection process was quite successful, resulting in 444 students with complete pretest data. We only used a subset of 265 students in our experiments because post-test data, used for extrinsic evaluation of our clustering, was available only for those 265 (the rest of the students either missed a tutoring session, or the post-test, or both).

It should be noted that the data is very diverse in terms of student prior knowledge of physics because students were recruited from a large variety of physics-related courses including introduction to physics, honors physics, and AP physics. This should allow us to draw general conclusions.

**Clustering Methods**

**DP-means algorithm**

The DP-means algorithm, as described by Kullis & Jordan (2012), is a hard-clustering approximation of nonparametric Bayesian models. Under the assumption that the DP-means is derived from a Dirichlet Process Mixture Model, there exists a lambda value $\alpha$ such that when
used by the algorithm, the number of clusters $k$ is identified. The DP-means algorithm is similar to the k-means clustering algorithm except that a new cluster is generated when the distance from a data point to the nearest cluster is larger than the threshold $\alpha$.

More specifically, the DP-means algorithm is derived from a Dirichlet Process Mixture Model (DPMM) as illustrated below:

- $\mu_1, \ldots, \mu_k \sim G_0$
- $\pi \sim Dir(k, \pi_0)$
- $z_1, \ldots, z_n \sim \text{Discrete}(\pi)$
- $x_1, \ldots, x_n \sim N(\mu_{z_i}, \sigma I)$
- The Dirichlet prior of dim $k$ is placed using some $\pi_0$

where:

- $\mu$ is the mean of each of the clusters, drawn from some base distribution $G_0$, which is the prior distribution over the means.
- $\pi = (\pi_1, \pi_2, \ldots)$ corresponds to the vector of probabilities of being in a cluster.
- $z_i$ is an indicator of cluster assignment.
- $x_i$ is a data point.

The corresponding clustering algorithm is described in Figure 2. The input consists of data instances $x_1, \ldots, x_n$, where $x_i$ represents the vector of pre-test answer choices of the $i^{th}$ student. Since the pre-test contains 35 questions, each such response vector $x_i$ contains 35 entries corresponding to each answer choice picked by student $i$. The clustering algorithm begins by initializing a single cluster whose mean is the global centroid. Then, it initializes a set of cluster
indicators: \( z_i = 1 \) for all \( i = 1, \ldots, n \) where \( z_i = k \) means that the student \( x_i \) belongs to the \( k^{th} \) cluster as denoted by \( l_k \).

In step 3, the algorithm computes the distances between each data point and the existing centroids. It then compares the minimum of these distances with \( \alpha \). If the minimum is larger than the threshold \( \alpha \), a new cluster is generated, and its centroid is assigned the current data point \( x_i \). Otherwise, the cluster indicator of the current data point is set to the \( argmin \) of the distances. After looping over all data points, the number of clusters \( k \) and the clusters indicators are computed. Finally, the DP-means algorithm generates the clusters \( l_j \) and their centroids \( \mu_j \) for \( j = 1, \ldots, k \). Step 3 is repeated until the algorithm converges.

**Algorithm: DP-means**

**Input:** \( x_1, \ldots, x_n \) : input data, \( \alpha \) : cluster penalty parameter  
**Output:** Clustering \( l_1, \ldots, l_k \) and number of clusters \( k \) 

1. Init. \( k = 1, l_1 = \{x_1, \ldots, x_n\} \) and \( \mu_1 \) the global mean. 
2. Init. Cluster indicators \( z_i = 1 \) for all \( i = 1, \ldots, n \)  
3. Repeat until convergence  
   - For each point \( x_i \)  
     - Compute \( d_{ic} = ||x_i - \mu_c||^2 \) for \( c = 1, \ldots, k \)  
     - If \( d_{ic} > \alpha \), set \( k = k + 1, z_i = k, and \ \mu_k = x_i \)  
     - Otherwise, set \( z_i = argmin_c d_{ic} \)
   - Generate clusters \( l_1, \ldots, l_k \) based on \( z_1, \ldots, z_k : l_j = \{x_i | z_i = j\} \)  
   - For each cluster \( l_j \), compute \( \mu_j = \frac{1}{|l_j|} \sum_{x \in l_j} x \).

**Figure 2.** DP-means algorithm

**K-modes Clustering**

K-modes clustering algorithm (Huang et al., 1998) is an extension of the k-means for the categorical data by using: (i) - a matching dissimilarity measures for categorical data points,(ii)- modes instead of means for clusters, and (iii)-a frequency based method to update
modes. The mean features of the k-modes clustering are its simplicity and easy implementation and efficiency in handling large of data objects. Its main issues are: (i) the need to define the number of clusters in advance, (ii) it is handling the categorical data only and (iii) producing the local optimum solutions.

Let X and Y be two categorical data objects with m attributes. The dissimilarity measure between X and Y is defined as following:

\[ d(X,Y) = \sum_{j=1}^{m} \partial(x_j, y_j) \]

Where:

\[ \partial(x_j, y_j) = \begin{cases} 
0 & (x_j = y_j) \\
1 & (x_j \neq y_j) 
\end{cases} \]

Let S be a set of categorical objects described by m categorical attributes \( A_1, ..., A_m \). A mode of \( S = \{X_1, X_2, ..., X_n\} \) is a vector \( Q = [q_1, ..., q_m] \) that minimizes:

\[ D(S, Q) = \sum_{i=1}^{n} d(X_i, Q) \]

Where Q is not necessarily an object of S.

The optimization problem for partitioning a set of n objects by m categorical attributes into k clusters \( S_1, ..., S_k \) becomes:

\[ \text{Minimize } \sum_{i=1}^{k} \sum_{X \in S_i} d(X, Q_i) \]
Where \( Q_i \) is the mode of the cluster \( S_i \).

The k-modes clustering algorithm is described in Figure 3.

<table>
<thead>
<tr>
<th>K-modes Clustering Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Data objects ( X ), Number of clusters ( K ).</td>
</tr>
<tr>
<td>2. Randomly select the ( K ) initial modes from the data objects such that ( C_j, j = 1, 2, \ldots, K ).</td>
</tr>
<tr>
<td>3. Find the matching dissimilarity between each ( K ) initial cluster modes and each data objects using the Eq. (2.1).</td>
</tr>
<tr>
<td>4. Evaluate the fitness using the Eq. (2.4).</td>
</tr>
<tr>
<td>5. Find the minimum mode values in each data object i.e. finding the objects nearest to the initial cluster modes.</td>
</tr>
<tr>
<td>6. Assign the data objects to the nearest cluster centroid modes.</td>
</tr>
<tr>
<td>7. Update the modes by applying the frequency-based method on newly formed clusters.</td>
</tr>
<tr>
<td>8. Recalculate the similarity between the data objects and the updated modes.</td>
</tr>
<tr>
<td>9. Repeat the step 4 and step 5 until no changes in the cluster ship of data object.</td>
</tr>
</tbody>
</table>

Figure 3. K-modes Clustering Algorithm

Experiments

Several experiments have been conducted to evaluate the performance of the proposed clustering algorithms in grouping students based on their prior knowledge. The dataset, the experimental setup and the obtained results are described next.

Dataset

The data used in our experiments consists of pre-test answers collected from 264 high-school students who took the aFCI pre-test, went through a 5-week training period, and then took a post-test. Furthermore, after each training sessions students took a short post-test (6 questions). In all our experiments, we will use this post-test after the very first training session as the extrinsic evaluation criterion as it is closest in time (among all post-tests) to the pre-test and therefore is a good estimate of students’ early knowledge states as best captured by the pre-test. The pretest includes 35 multiple choice questions that have the same weight. Two types of data have been used in our experiments: 5-way response data and binary response data. The categorical data
consists of the actual answer choices students picked for the 35 multiple choice questions coded as A, B, C, D and E. For each question, one those choices is the correct answer. The binary data represents the same data coded as binary correctness values: 0 – incorrect, i.e., the student picked any of the incorrect answer choices, and 1 – correct, i.e., the student picked the correct answer choice.

Tables 2 and Table 3 illustrate the data representation for the two tables. As described below, the columns represent the 35 questions, and the rows represent individual students’ responses.

Table 2. Categorial Data

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>…</th>
<th>Q35</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dh001</td>
<td>A</td>
<td>B</td>
<td>…</td>
<td>C</td>
</tr>
<tr>
<td>Dh002</td>
<td>C</td>
<td>D</td>
<td>…</td>
<td>C</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>DH356</td>
<td>C</td>
<td>D</td>
<td>…</td>
<td>B</td>
</tr>
</tbody>
</table>

Table 3. Binary Data

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>…</th>
<th>Q35</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dh001</td>
<td>0</td>
<td>0</td>
<td>…</td>
<td>0</td>
</tr>
<tr>
<td>Dh002</td>
<td>1</td>
<td>0</td>
<td>…</td>
<td>1</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>DH356</td>
<td>1</td>
<td>0</td>
<td>…</td>
<td>0</td>
</tr>
</tbody>
</table>
Experiments: Binary Data

A first set of experiments have been conducted using the binary data as input for the DP-means algorithm. Since we have binary data and DP-mean is based on the Euclidean distance, we have applied Principal Component Analysis (PCA) to convert the binary values to continuous ones. For this purpose, numerous values of n (number of components) have been tested. The value 35 led to a convergence state of 10 clusters in which several clusters are redundant, i.e., using the extrinsic criterion based on the overall post-test score. For example, the average of the post test score for clusters 6, 7 and 9 is 3.0. Thus, we have tested randomly several values. The value 24 led to better clustering results in terms of splitting well the clusters based on the extrinsic criterion. Thus, we used those components to represent our data points for the rest of the experiments. On the binary data, a Manhattan distance could be used which we tried and didn’t lead to better results than the above method of using PCA.

The $\alpha$ distance parameter has not been defined a priori. To select a suitable value of this parameter, we followed first the procedure described by Kulis and colleagues (2012) as in the following: given k=3 as the desired number of clusters, we first initialize a set A with the global mean of the data. Then iteratively we calculate the maximum distance to A (the distance to A is the smallest distance among points in A). We repeat this k (=3) times and assign to $\alpha$ the value of the maximum distance to A. In our work, we got the value of 3.26. Testing the DP-means with this value led to the convergence of two clusters of students. To reach the desired number of clusters which is 3, we have tried other values in a close interval of $[2.9, 3.1]$ as described in Table 5.

Thus, various values have been tested and compared. The evaluation of the resulted clusters has been done using the following measures:
- **Silhouette index**: Its value measures how similar a student response vector (her set of responses to the pre-test questions) is to its own cluster (cohesion) relative to the other clusters (separation). The silhouette index is a value within the \([-1, +1]\) interval. A high value of the silhouette index indicates that the student is well matched with the other students in the same cluster. The following metric distances have been tested: Euclidean distance, Manhattan distance and Cosine similarity. The obtained results have shown that the Euclidean distance led to better results as demonstrated in Table 4.

- **Mean of post test score**: The data collected from the interaction of the students with the ITS includes post test scores for the 264 students. Since the post test is taken at the end of the experiment, weeks after the students took the pre-test, and since it has not been used in the cluster, it can be used as an extrinsic measure of cluster validity and interpretation of the resulting clusters. Indeed, this measure is used by us to assess the mastery level of each resulted cluster of students. In addition, it has been used as a way to check the separation of the clusters. The maximum and minimum values of the post test score in this collected data are 6.0 and 0.0 respectively.

- **Mean of pretest score**: The data collected includes the pretest performance of each student based on the correct answers. The highest value is 35 and the lowest value is 0.
Table 4. Clustering results of DP-means with different types of distances

<table>
<thead>
<tr>
<th>Distance</th>
<th>$\alpha$</th>
<th>Number of clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manhattan</td>
<td>2.9</td>
<td>255</td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>255</td>
</tr>
<tr>
<td></td>
<td>3.1</td>
<td>255</td>
</tr>
<tr>
<td>Euclidian</td>
<td>2.9</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3.1</td>
<td>2</td>
</tr>
<tr>
<td>Cosine</td>
<td>2.9</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3.0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3.1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5. DP-means clustering results with different values of $\alpha$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Clusters</th>
<th>Mean pretest score</th>
<th>Mean post-test score</th>
<th>Number of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.9</td>
<td>C1</td>
<td>15.26</td>
<td>3.31</td>
<td>207</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>31.28</td>
<td>5.66</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>C3</td>
<td>6.47</td>
<td>1.89</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>C4</td>
<td>25.0</td>
<td>5.0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>C5</td>
<td>14.0</td>
<td>2.0</td>
<td>1</td>
</tr>
<tr>
<td>3.0</td>
<td>C1</td>
<td>17.68</td>
<td>3.44</td>
<td>195</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>31.11</td>
<td>5.64</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>C3</td>
<td>8.83</td>
<td>1.62</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>C1</td>
<td>13.87</td>
<td>3.18</td>
<td>277</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>29.54</td>
<td>5.64</td>
<td>37</td>
</tr>
</tbody>
</table>

Figure 4. Quality of the DP-means algorithm using different values of $\alpha$
The results in Figure 4 show that the value 3.0 of parameter $\alpha$ led to the highest value of the Silhouette index (0.27). In addition, this $\alpha$ value resulted in three distinct clusters, well separated (Figure 4), in terms of students’ performance in the post test and pretest (as described in Table 5). The first cluster contains 195 students. The mean post test score is 3.44 and the mean pretest score is 17.68 which are average scores. Students who belong to this cluster can be described as average performers or learners. The second cluster contains 37 students. The mean post test score is 5.66 and the mean pretest score is 31.11 which are high scores. The students in this cluster can be described as high performers or learners of Physics. The third cluster contains 32 students. The mean post test score is 1.625 and the mean pretest score is 8.83 which are very low scores. The students of this cluster can be describing as struggling ones.

Figure 5. DP-means visualization with $\alpha = 3$

Figure 6. DP-means visualization with $\alpha = 3.1$
Figure 7. DP-means visualization with $\alpha = 3.2$

To compare the results of the DP-means algorithm with other clustering algorithms, we have also applied the k-means and agglomerative clustering algorithms on the same binary data. Since the best result of the DP-means was for an $\alpha$ value 3.0, we ran the k-means algorithm using $k = 3$ and the agglomerative clustering using the same number of clusters (=3). Table 6 and Table 7 present the results for k-means and agglomerative clustering, respectively.

### Table 6. k-means results

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Mean Post-Test Score</th>
<th>Mean Pretest Score</th>
<th>Number of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>2.28</td>
<td>9.44</td>
<td>97</td>
</tr>
<tr>
<td>C2</td>
<td>5.21</td>
<td>28.84</td>
<td>52</td>
</tr>
<tr>
<td>C3</td>
<td>3.83</td>
<td>17.22</td>
<td>115</td>
</tr>
</tbody>
</table>

### Table 7. Agglomerative clustering results

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Mean Post-Test Score</th>
<th>Mean Pretest Score</th>
<th>Number of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>3.55</td>
<td>16.46</td>
<td>165</td>
</tr>
<tr>
<td>C2</td>
<td>5.45</td>
<td>30.0</td>
<td>42</td>
</tr>
<tr>
<td>C3</td>
<td>2.07</td>
<td>8.28</td>
<td>57</td>
</tr>
</tbody>
</table>
Figure 8. Quality of DP-means algorithm using different values of $\alpha$.

The results depicted in Table 5 show that the DP-means algorithm with $\alpha=3.0$ outperforms the k-means and the agglomerative algorithms as described in tables 6 and 7 respectively. The difference, in the mean post test score and the mean of pretest score, between the clusters of DP-means is larger than the difference using the other clustering algorithms. This indicates the convergence of well separated groups of students, in terms of learning level and prior knowledge, when applying the DP-means.

A detailed analysis of the top 10 students closest to the centroids of each of the three clusters found by DP-means, revealed that students in cluster 1 struggled mostly with questions related to Newton’s third and first laws, whereas students in cluster two struggled with questions related to Newton’s third law. Students in cluster 3 struggled the most and they showed weaknesses across all major topics in Newtonian Physics. Since in this experiment we used just correctness values for each pre-test question it is not possible to provide a more detailed analysis in terms of specific misconceptions, e.g., assuming faster velocity implies a larger force in an action-reaction pair, students in each clusters exhibit.
Experiments: Clustering Categorical Data

A second set of experiments have been conducted using categorical data, DP-means and k-modes clustering algorithms. That is, in this case, we used the actual answer choices picked by students for the pre-test questions to find the clusters.

To this end, first, we have converted the categorical responses to numerical ones using one-hot encoding. Basically, each answer choice becomes a dimension in a vector space representation. A value of 1 is assigned to that dimension for a given question in the pre-test if a student picked the choice corresponding to the dimension as their answer choice. This, results in an encoding of categorical integer features as a one-hot numeric array. The encoder derives the categories based on the unique values in each feature. The output of the one-hot-encoding is fed into the clustering algorithms.

The results presented in Table 8 reflect a decrease in quality of the DP-means clustering using categorical data. The silhouette index, as described in Figure 8, has decreased in comparison with the DP-means based on binary data. The highest value was 0.06 when using the value 4.4 of $\alpha$.

The different values of $\alpha$ didn’t lead to a good split of students in terms of the performance. For example, in the case of $\alpha = 4.4$, cluster C2 and C3 can be merged in one cluster since their mean post-test and pretest scores are very close. For $\alpha = 4.3$, there is redundancy in the resulted clusters. For example, C3 and C6 can be merged in one cluster.
Table 8. DP-means clustering results (categorical data) with different values of $\alpha$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Clusters</th>
<th>Mean Post Test Score</th>
<th>Mean Pretest Score</th>
<th>NB of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>C1</td>
<td>4.17</td>
<td>20.39</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>1.5</td>
<td>5.5</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td>C3</td>
<td>1.0</td>
<td>10.0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>C4</td>
<td>2.0</td>
<td>4.5</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>C5</td>
<td>2.0</td>
<td>6.0</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>C6</td>
<td>1.66</td>
<td>6.0</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>C7</td>
<td>2.16</td>
<td>9.35</td>
<td>12</td>
</tr>
<tr>
<td>4.4</td>
<td>C1</td>
<td>4.12</td>
<td>20.14</td>
<td>187</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>1.0</td>
<td>10.0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>C3</td>
<td>1.66</td>
<td>6.0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>C4</td>
<td>2.15</td>
<td>9.0</td>
<td>73</td>
</tr>
<tr>
<td>4.5</td>
<td>C1</td>
<td>3.55</td>
<td>16.87</td>
<td>263</td>
</tr>
<tr>
<td></td>
<td>C2</td>
<td>1.0</td>
<td>10.0</td>
<td>1</td>
</tr>
</tbody>
</table>

To overcome this drawback of DP-means when applied to categorical data, we have applied the k-modes clustering algorithm (Huang et al., 1997; Huang et al., 1998). The k-modes algorithm is based on defining the dissimilarity measure between objects. This dissimilarity between two objects A and B can be defined by the total mismatches of the corresponding attribute categories of the two objects. The smaller the number of mismatches is, the more similar the two objects.

The following are the results with k-modes when using $k = 3$.

Table 9. K-modes results with $k = 3$

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Mean Post Test Score</th>
<th>Number of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>3.72</td>
<td>120</td>
</tr>
<tr>
<td>C2</td>
<td>5.03</td>
<td>59</td>
</tr>
<tr>
<td>C3</td>
<td>2.23</td>
<td>85</td>
</tr>
</tbody>
</table>

The results listed in Table 9 demonstrate that the k-modes outperforms the DP-means when using categorical data. The resulted clusters reflect a good split between clusters in terms of
performance in the post test. The C1 cluster reflects an average knowledge level of students. C2 reflects a high level of knowledge of students. And C3 reflects a low level of learning. A more detailed analysis indicates the same overall conclusions reached using the correctness data, e.g., students in cluster one struggle mostly with Newton’s second and third laws. However, when using the categorical data, we can further pinpoint which aspects of Newton’s third law for instance, students struggle with. For instance, many students in cluster C1 seem to struggle with the misconception that in an interaction between two objects the more massive one will act with a bigger force on the smaller one which is not true. According to Newton’s third law, to each action there is an equal and opposite reaction. Therefore, this analysis suggests that when a new student uses a Physics ITS, after they take the pre-test and their answer patterns place him closer to the centroid of cluster C1, i.e., in cluster C1, then appropriate instructional tasks that have been designed for students in that cluster should be activated in order to overcome major gaps students in that cluster exhibit.

**Conclusions**

In this work, DP-means clustering algorithm has been applied on the pretest data of 264 students collected from their interaction with DeepTutor ITS. Various values of $\alpha$ have been tested. The results demonstrated that 3.0 is the best value and three distinct clusters of students have been converged. These clusters reflect three distinct levels of learning which has been assessed using post test scores. The first cluster of students correspond to an average level of learning, the second cluster represents students with a high level of learning and the third cluster of students those with a low level of learning. Results have demonstrated also that DP-means outperforms k-means and Agglomerative clustering in terms of splitting well students based on their performance in the post test. Another finding is that the quality of DP-means algorithm, measured by the silhouette index,
decreases when we use the categorical data in comparison with the binary data. To overcome this drawback, we have used the k-modes clustering.

Furthermore, such clustering could offer a good trade-off between adaptivity and authoring costs. For instance, macro-adaptation can be expensive if the number of unique student knowledge states is very large as it requires selecting a unique set of tasks for each such unique knowledge state. Concretely, if using a 5-way/choice 35 multiple-choice question pre-test, the number of possible combinations of 35 answers is $5^{35}$, an extremely large space. That is, if each student’s knowledge state is described by the 35 responses, we end up with $5^{35}$ student knowledge states or student models which, by comparison, is much larger than the world’s population which is a bit over 514. Considering each of these potential knowledge states and selecting for each corresponding learner a unique set of tasks becomes a computationally and authoring challenge. An alternative, for instance, would be to group students into clusters of similar mental models and then select and author tasks for each such clusters. That is, grouping students into similar mental model groups can offer a good trade-off between adaptivity and authoring costs.

We plan to further investigate the resulting clusters for a better understanding of the characteristics of the students in each cluster. For instance, we do have information about the Physics class (intro, honors, AP) each student took and therefore a detailed analysis for students in each cluster based on their class type can be performed to understand what the major misconceptions students in each class struggle are with. Not only this could inform an ITS for Physics, but this information can be shared with teachers to help them better plan their lessons plans to address major misconceptions their students may have.
Chapter 3

Assessment of Open-Ended Student Answers Using Bi-GRU Capsule Networks

Introduction

Automatically assessing open-ended short student responses is an extremely challenging task due to the variability of the natural language, the reference matching problem. For instance, students can express their responses in numerous ways owing to different individual styles and varied cognitive abilities and knowledge levels. This assessment plays a vital role in improving the learning process as the system provides hints and feedbacks for the struggling students with incorrect answers.

The widely adopted and scalable approach to assessing such open-ended student responses is semantic similarity which is a complex NLP task as the meaning of words changes significantly when the context is changed. As Jiang (1997) quotes, “In many cases, humans have little difficulty in determining the intended meaning of an ambiguous word, while it is extremely difficult to replicate this process computationally”. According to the semantic similarity approach, a score, usually normalized, is computed between a target student answer and an expert-provided reference answer (Banjade et al., 2016). If the student answer has a high semantic similarity score to the reference answer, we infer that the student answer has the same correctness value as the reference answer. A low semantic similarity score implies the student response is incorrect. It should be noted that sometimes the reference answer may denote a common misconception in which case a high-similarity score to such misconception means the student answer also indicates a misconception, i.e., it is incorrect.
More broadly, the task of computing the semantic similarity of two texts consists of determining, both quantitatively (e.g., normalized score between 0 and 1) or qualitatively (are the two texts in a paraphrase, elaboration, or entailment relation) the degree of similarity between the two texts. It is a widely used step in many natural language processing (NLP) applications such as text summarization (Wong et al., 2008; Nenkova et al., 2011), question answering (Vo et al., 2015) and machine translation (Corley and Mihalcea, 2005). It should be noted that we can distinguish among semantic similarity tasks and methods at various granularity levels: word-to-word similarity (w2w), phrase-to-phrase (ph2ph), sentence-to-sentence (s2s), paragraph-to-paragraph (p2p), and document-to-document (d2d) similarities.

Several approaches have been proposed to automatically assess the semantic similarity of short sentence-level texts, which are our focus. Deep learning has recently attracted a wide attention in NLP field. This approach has the advantage of not needing hand-crafted features and other external resources, that is, just the raw sentences and the corresponding pre-trained word embeddings are needed as input. To this end, numerous deep learning methods have reached the state-of-the-art results. For example, Pontes and colleagues (2018) proposed a deep learning model that combines convolution and recurrent neural networks to measure the semantic similarity of sentences. This combination of networks has been helpful in capturing the most relevant information of sentences. Thus, improving the computation of semantic similarity scores relative to state-of-the-art systems. Other approaches worth-mentioning are: (1) the nonlinear similarity approach (Tsubaki et al., 2016), where word representations are inferred through the similarity learning of sentences in high-dimensional space with kernel functions, (2) constituency tree LSTM (Tai et al., 2015) which is a generalization to LSTMs to tree-structured network topologies, and (3) skip-thought (Kiros et al. 2015), where an encoder-decoder model is used to reconstruct the
surrounded sentences. Then, sentences with common semantic and syntactic properties are mapped to similar vector representations. Furthermore, Bao et al. (2018) proposed an Attention Siamese Long Short-Term Memory (LSTM) model to measure the semantic textual similarity. An attention mechanism has been used to capture the high-level semantic information. The empirical experiments have demonstrated the effectiveness of the model with an impressive performance. Wang and colleagues (2018) presented an approach that combines a Bidirectional Long Short-Term Memory Networks (BLSTM) and Convolutional Neural Networks (CNN) to extract the semantic features of a sentence. Then sentence representations are learned with word-level attention. Finally, an output layer that calculates the similarity score was used. This proposed model was evaluated using the Quora Duplicate Questions Public dataset. The obtained results showed that this model has outperformed many existing approaches, such as Support Vector Machine (SVM), CNN, BLSTM and Attention based BLSTM, with a highest accuracy of 0.89.

Our approach is very different from these reviewed approaches except the fact that uses the deep learning approach.

Our task of automatically assessing freely generated student answers within a dialog system context is a special case of the more general semantic similarity task. As shown in Figure 9, given two inputs, the student answer and the reference answer, the assessment model computes the correctness of the student answer. Typically, the reference and student answer are domain specific as tutoring targets specific science topics, e.g., Physics. Furthermore, the answers are generated in the context of problem-solving instructional activities in which students are asked to provide solutions to various problems in the form of short essays, the essays are evaluated and if incorrect and/or incomplete a tutorial dialogue follows in which students provide short answers to
Motivated by the good results of deep learning models in similar semantic similarity task, we present in this dissertation a Bi-GRU Capsule Neural Networks model to assess the students’ answers generated during student-system dialogue-based interactions. Capsules have the capability to encode the semantic meanings in a wider space using a vector. Thus, these capsules are suitable to express a sentence as a vector (Kim. J et al.,2018). This generated vector captures the instantiation parameters of the input such as the order of the words and their semantic representation. On the other hand, word embeddings also transform words into lower dimensional vectors that preserve the contextual similarity of words. In general, the embedding vectors are fed into various deep learning models. Our model consists of several important components. First, an embedding layer that transforms each word of the input to a distributed vector representation. Second, the resulted embedding matrix is imported into a Bidirectional Gated Recurrent Units layer (Bi-GRUs) (Cho et al., 2014) to encode the input text into a fixed length representation. The fixed length representation is then fed into a capsule layer. Finally, the capsule layer’s outputs are fed to a fully connected dense layer with SoftMax activation for classification. We evaluated the performance of our model using the DT-Grade corpus (Banjade et al., 2016).
Related Prior Work

Capsule networks have been introduced by Geoffrey Hinton for image classification to overcome the limitations of the Convolution Neural Networks particularly in the pooling layer that causes the loss of important information regarding the spatial relationships. These networks are based on so called capsules and are trained using a dynamic routing algorithm (Sabour et al., 2017). Each capsule encodes a particular feature (e.g. local order of words, semantic representations of words) that the network is looking for. The magnitude of a capsule vector defines the probability of the existence of that feature. The layers of capsule networks are connected via computing a learned vector between each pair of capsules. Then, the routing algorithm is used to ensure that the output of the capsule, which is a vector, gets sent to an appropriate parent in the layer above. The capsule computes a “prediction vector” for each possible parent. This prediction vector is calculated by multiplying the capsule ‘s own output by a weight matrix. A top-down feedback is applied, in case the prediction vector has a large scalar product with the output of a possible parent. This is done to increase the coupling coefficient for that parent and decrease it for other parents. In sum, this iterative routing process decides the credit attribution between the nodes in lower and higher levels. Recently, several NLP researchers have applied Capsule Networks for various tasks such as text classification and sentiment analysis. The obtained results were very impressive and encouraging to further investigate these networks in related tasks.

Capsule Networks showed a competitive performance in the text classification task. For example, Zhao and colleagues (2018) used Capsule Networks with dynamic routing algorithm for text classification. To boost performance, they have applied three different strategies to stabilize the dynamic routing process by decreasing some noise capsules. First, an Orphan category has been added to the network to capture the background information such as stop words and the words
that are unrelated to specific categories. Second, a Leaky-SoftMax approach has been used to update the connection strength between the parent capsules and their children. Third, the connection strength has been amended using the probability of the existence of the child capsules. To evaluate the performance of the proposed approach, they have conducted several experiments using six different datasets. The obtained results demonstrated the superior performance of Capsule Networks over many baseline methods. Our approach is similar in the sense that we model the student answer assessment task as a text classification task. However, the architecture of our proposed model is different. In fact, Zhao and colleagues’ model consists of a convolutional layer after the embedding layer and our proposed model consists of a Bi-GRU layer instead. In continuous research efforts, Kim and colleagues (2018) have applied Capsule Networks for text classification as well. They have used a simple dynamic routing algorithm to boost the efficiency of the model. Their proposed model consists of the following components: (1) an embedding layer, (2) a feature map that use convolutions, (3) a convolutional capsule layer, and (4) a text capsule layer. The authors have conducted several experiments using different datasets. The experimental results demonstrated the potential of the Capsule Networks in the text classification task. This approach is similar to our work in the sense of considering the student answer assessment task as a text classification task. The main difference is using a Bi-GRU layer instead of convolutions after the embedding layer.

Capsule Networks have been applied successfully in other NLP tasks. For example, Zhang and colleagues (2018) proposed a relation extraction approach based on Capsule Networks with attention mechanism. Wang and colleagues (2018) presented an Attention-based Bi-GRU-Capsnet model to detect hypernymy relationship between compound entities. Xia and colleagues (2018) proposed two capsule-based architectures to solve the zero-shot intent detection problem: the
INTENT-Capsnet that extracts semantic features from utterances and aggregate them to discriminate existing intents, and INTENT-Capsnet to discriminate emerging intents via knowledge transfer from existing intents.

Based on these successes of Capsule Networks on related tasks, we have explored their potential for assessing student answers task. To the best of our knowledge, this is the first attempt at using Capsule Networks (at the time the publication) for assessing student generated answers in conversational intelligent tutoring systems.

**Model Architecture**

Our proposed model (Figure 10) consists of four major components: (1) an embedding layer that transforms each word to a distributed vector with a dimension $d$, (2) a bidirectional- GRU encoder, (3) a Capsule Network that generates high level semantic representations of the student and reference answers using a dynamic routing algorithm, (4) a SoftMax layer that computes the probabilities of the correctness classes.

**Embedding Layer**

Given a student answer $X$ and a reference answer $X'$, we tokenize them into a sequence of words: $X = [w_1, \ldots, w_n]$ and $X' = [w'_1, \ldots, w'_m]$. Afterwards, each token is converted into a $d$-dimensional vector through the embedding layer. In this work, we consider the following word embeddings approaches: GloVe, Word2vec and ELMo.

- GloVe embedding has been proposed by Pennington et al. (2014). It is a “count b-based” model where the word co-occurrence count matrix is pre-processed by normalizing the counts and log-smoothing operation. This matrix is then factorized to get lower dimensional representations. GloVe model was trained using five different
corpora mostly Wikipedia. The GloVe loss function minimizes the least-square distance between the context window co-occurrence values and the global co-occurrence values (Juan et al., 2019).

- Word2vec embedding has been proposed by Mikolov and colleagues (2013). Developed from Google News dataset containing approximately 3 million vector representations of words and phrases, word2vec is a neural network model used to produce distributed vector representation of words based on an underlying corpus. Two models have been proposed: CBOW and skip-gram. CBOW computes the probability of a target word given the context surrounding words within a window. Skip-gram is the opposite of CBOW model where the probability of the surrounding words is computed given the target word.

- ELMo (Peters et al., 2018) method produces word embeddings for each context where the word is used, thus allowing different representations for the same word. The mechanism of ELMo is based on the representation obtained from a bidirectional LSTM model (BiLSTM). It consists of two language models: forward LSTM and backward LSTM. When training a deep BiLSTM, the higher-level LSTM states capture context-dependent aspects of the word meaning, while lower-level LSTM states capture the aspects of syntax. ELMo encodes all this information in a single word vector. It should be noted that the use of ELMo embedding has boosted significantly the performance of several deep learning models.
Figure 10. Bi-GRU-CapsNET model architecture: it consists of the following layers : (i) embedding layer, (ii) Bi-GRU layer, (iii) Capsule Layer and (iv) SoftMax Layer

**Bi-GRU Layer**

A Gated Recurrent Unit (GRU) model is a variant of the recurrent neural network (RNN). GRU has two gates: an update gate z and a reset gate r. The update gate determines how much memory of previous cell to keep alive, and the reset gate determines how to combine the input of new cell with the previous memory. For each position $t$, GRU computes $h_t$ with input $x_t$ and previous state $h_{t-1}$, as:

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$
\[ u_t = \sigma(W_t x_t + U_u h_{t-1}) \]

\[ \tilde{h}_t = \tanh(W_c x_t + U(r_t \cdot h_t - 1)) \]

\[ h_t = (1 - u_t) \cdot h_{t-1} + u_t \cdot \tilde{h}_t \]

Where \( h_t \), \( r_t \) and \( u_t \) are d-dimensional hidden state, reset gate, and update gate. \( W_r, W_u, W_c \) and \( U_r, U_u \) and \( U \) are the parameters of the GRU model. \( \sigma \) is the sigmoid function, and \( . \) is the element-wise production.

The outputs vectors \( h_t \) and \( h_t' \) are fed into the capsule layer.

**Capsule Layer**

The assumption behind Capsule Networks is that there are capsules (group of neurons) that tell whether certain entities are present in an image (text in our task). A capsule as shown in Figure 11 has an activation vector that represents the instantiation parameters of an entity and whose length represents the probability of the existence of that entity.

![Figure 11. Capsule structure](image-url)
Given the input vectors $u_1, u_2$ and $u_3$ from the previous layers, a learned transformation matrix $W_{ij}$ is applied to generate the predictors vectors $\hat{u}_i$ as following:

$$\hat{u}_{j|i} = W_{ij}u_i$$

Then, in the higher layer, a capsule $s_j$ is computed by the linear combination of all the prediction vectors with weights $c_{ij}$ as following:

$$s_j = \sum_i c_{ij}u_{j|i}$$

where $c_{ij}$ are coupling coefficients computed by the dynamic routing algorithm described in Figure 12.

---

**Routing Algorithm**

1: procedure ROUTING ($\hat{u}_{ji}, r, l$)
2: for all capsule $i$ in layer $l$ and capsule $j$ in layer $l + 1$:
   $b_{ij} \leftarrow 0$
3: for $r$ iterations do
4:    for all capsule $i$ in layer $l$:
   $c_i \leftarrow softmax(b_i)$ SoftMax computes Eq.3
5:    for all capsule $j$ in layer $(l + 1)$:
   $s_j \leftarrow \sum_i c_{ij}\hat{u}_{j|i}$
6:    for all capsule $j$ in layer $(l + 1)$:
   $v_j \leftarrow squash(s_j)$ squash computes Eq.1
7:    for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l + 1)$:
   $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} \cdot v_j$
   return $v_j$

Figure 12. The Routing Algorithm
As stated before, the output of a capsule represents the probability that the input has the entity that the capsule describes. So, the range of the activation vector should be in the [0,1] interval. For this purpose, a squash function is applied to generate the final output vector $v_j$ as following:

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\| \|s_j\|^2}$$

The final outputs of the capsule layers for the given students’ answers are activation vectors $v_1$ and $v_2$. Afterwards, we concatenate these vectors as $[v_1, v_2]$ and we feed this concatenation into a SoftMax layer that computes the probability for each correctness class.

**Experiments**

We test the proposed Bi-GRU-Capsnet model using different parameters settings and embedding approaches. We evaluate the performance of the proposed model using the DT-Grade dataset as described next.

**DT-Grade Dataset**

The DT-Grade dataset (Banjade et al., 2016) was created by extracting student responses from logged tutorials interactions between 36 junior level college students and a state of the art ITS. During the interactions, each student solved 9 conceptual Physics problems – they had to provide the correct answer and a full justification based on Physics principles. Their answer was evaluated and if the answer was incorrect or incomplete, e.g., a partial as opposed to full justification was provided, a dialogue followed in which the ITS helped the student discover the correct solution through personalized scaffolding in the form of hints that varied in their degree of information/help.
provided depending on students’ needs. In case the student provided a correct but incomplete answer, the goal of the dialogue was to elicit the missing parts of the solution. Eliciting the full justification from the student reveals their thinking and whether they found the right answer through correct or incorrect reasoning. When detected, misconceptions are immediately corrected.

Each instance in the DT-Grade dataset includes the following components: (1) the Physics problem description, (2) the tutor question, (3) the student answer (as typed by the students, i.e., without correcting spelling or grammatical errors) and (4) reference/ideal answer(s). Each student response was assessed by a team of human experts and categorized into one of the following four correctness classes:

1. **Correct**: Answer is correct
2. **Correct-but-incomplete**: The response provided by the student is correct, but something is missing.
3. **Incorrect**: Student answer is incorrect.
4. **Contradictory**: The student answer is contradicting the reference answer.

Table 10. Annotation example of the DT-Grade dataset

<table>
<thead>
<tr>
<th><strong>Problem Description</strong>:</th>
<th>A car windshield collides with a mosquito, squashing it.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Question</strong>:</td>
<td>How does Newton's third law apply to this situation?</td>
</tr>
<tr>
<td><strong>Student Answer</strong>:</td>
<td>the windshield will apply a force to the mosquito equal the force applied by the mosquito to the windshield</td>
</tr>
<tr>
<td><strong>Reference answer</strong>:</td>
<td>1: Since the windshield exerts a force on the mosquito, which we can call action, the mosquito exerts an equal and opposite force on the windshield, called the reaction.</td>
</tr>
</tbody>
</table>
In this work, we consider only two classes: correct and incorrect. The mapping from the original DT-Grade labels to ours was done as described next. The correct answers are those labeled as “correct” in the DT-Grade dataset. All the other instances are considered “incorrect”. As a result, we obtained the following class distribution shown in Table 11 below.

Table 11. The distribution of classes in training (800 instances) and testing data (100 instances)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Correct (%)</th>
<th>Incorrect (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>41</td>
<td>59</td>
</tr>
<tr>
<td>Testing</td>
<td>41.58</td>
<td>58.41</td>
</tr>
</tbody>
</table>

**Experimental Setting**

Several experiments have been conducted with different capsule neural networks parameters settings varying the embedding representations and the number of capsules to evaluate the performance of our proposed model using the DT-Grade dataset.

A first set of experiments have been conducted using the pretrained Glove embeddings with 100 dimension and three different values of the number of capsules. Based on the literature, we have started with a value of 10 and added two other values: 15 and 20. This has been done to test the impact of different expressiveness levels of the capsule network layer on the performance. A second set of experiments have been conducted using word2vec embeddings with 100 dimensions while using the same different values of the number of capsules. Another set of experiments have been run using ELMo embeddings with 300 dimensions, which are the state of art of embeddings, while using the same values of the number of capsules. To compare the performance of our model with existing ones, we have empirically experimented the following baseline deep learning
models: (1) An LSTM (Long Short-Term Memory) neural network that consists of a Glove embedding and 240 cells. (2) A Bi-GRU network that consists of Glove embedding with 50 units.

During the experiments, we used 80% of data set for training and 20% for testing. The distribution of classes, as shown in Table 3.2, in training and testing is imbalanced. To overcome this problem, we adjusted the class weights in the model during the training.

**Hyperparameters**

In all the experiments, we used a Bi-GRU layer with 50 units. Several numbers of units have been tested and this value has led to higher accuracy. We also added a 0.2 Dropout to the Bi-GRU layer to prevent over-fitting. For the capsule layer, we used 3 iterations for the routing algorithm and 16 for the capsule dimension. For optimization, we used the Adam optimizer (Kingma et al., 2014) with a learning rate of 0.0001. The gradients are clipped to 0.5 to prevent exploding gradients. We trained our model for 100 epochs to obtain the results. The increase of epochs, particularly when using the ELMo embedding, showed an increase in the overall accuracy and F1-measure values.

**Results & Discussion**

Table 12 shows the results on the DT-Grade dataset. Our Bi-GRU-Capsnet model outperforms the baselines deep learning models, particularly the Bi-GRU and LSTM models. The results show that our model reaches the highest accuracy of 72.5 and 0.7 of F1-measure when using the ELMo embedding yielding the start of the art results on the DT-Grade dataset. This is not a surprising result. Several research works have demonstrated that ELMo embeddings boosted the performance of deep learning models in various NLP tasks. However, the accuracy and F1 score decreased significantly when using the word2vec embedding approach. The obtained accuracy is considered
a very good result for the DT-Grade dataset due to its small size in comparison with larger NLP datasets.

Table 12. The performance results of various models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy %</th>
<th>F1 Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bi-GRU-capsnet (Glove,10)</td>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>Bi-GRU-capsnet (Glove,15)</td>
<td>60.62</td>
<td>55</td>
</tr>
<tr>
<td>Bi-GRU-capsnet (Glove,20)</td>
<td>58.75</td>
<td>60</td>
</tr>
<tr>
<td>Bi-GRU-capsnet (Word2vec,10)</td>
<td>55</td>
<td>59</td>
</tr>
<tr>
<td>Bi-GRU-capsnet (Word2vec,15)</td>
<td>56.25</td>
<td>57</td>
</tr>
<tr>
<td>Bi-GRU-capsnet (Word2vec,20)</td>
<td>52.25</td>
<td>47</td>
</tr>
<tr>
<td>Bi-GRU-capsnet (ELMo,20)</td>
<td>69.37</td>
<td>68</td>
</tr>
<tr>
<td>Bi-GRU-capsnet (ELMo,15)</td>
<td>66.25</td>
<td>66</td>
</tr>
<tr>
<td>Bi-GRU-capsnet (ELMo,10)</td>
<td>72.5</td>
<td>70</td>
</tr>
<tr>
<td>Bi-GRU(Glove)</td>
<td>56.25</td>
<td>56</td>
</tr>
<tr>
<td>LSTM(Glove)</td>
<td>60</td>
<td>60</td>
</tr>
</tbody>
</table>

Conclusions

Capsule Networks have demonstrated a good performance in several NLP tasks especially in text classification. Motivated by these successes, we proposed a Bi-GRU-Capsule Networks model to assess the correctness of the students answers within the state of art of intelligent tutoring systems DeepTutor. We have chosen this deep learning model to get benefits from its no requirements of hands-crafted features and external resources. Added to this, Capsule Networks have the capability to express the semantic meanings in a wider space using a vector that captures the instantiation parameters of the input such as the order of words and their semantic representation. The experimental results showed that our model achieved the state of the-art-results on the DT-Grade dataset surpassing the previous methods such as LSTM based model. Particularly, our model reached the highest accuracy when using the ELMo embeddings.

Despite the state of-the-art results, this proposed model was incapable to assess correctly the very short answers that contains very few words less than 5. The next proposed approach was
driven by overcoming this shortcoming. Therefore, we proposed a deep learning model that utilizes a contextual attention mechanism.
Chapter 4

An-Attention Based Transformer Model for Student Answers Assessment

Introduction

As stated in the previous chapters, student answers assessment is an important component in an intelligent tutoring system (ITS). Evaluating the knowledge state of students informs the adaptivity of the ITS and optimizes the learning gains of students. Detecting the struggling students with low performances makes the system provide the appropriate hints and feedbacks to find the missing steps in the solutions. Semantic similarity approach has been adopted widely by several researchers to tackle this problem. According to it, a similarity score between 0 and 5 is computed between a student answer and an expert answer. A high score implies the correctness of the answer, and low score implies the incorrectness of the student answer. This is a challenging task as students can use very short answers and pronouns to refer to some concepts in the previous question or problem description.

The attention mechanism in deep learning allows to pay greater attention to specific factors when processing the data. The use of attention in a neural network can results in a performance gain as the weight computed by attention could help in capturing the most relevant items in the input and discard the irrelevant ones. For these reasons, attention has become a frequent element of several neural architectures for NLP (Gatt & Krahmer, 2018; Young, Hazarika, Poria, & Cambria, 2018). The following example explains how the attention works on a higher level.

Given the following input sentence:

“The animal didn't cross the street because it was too tired”.

51
Does “it” refer to animal or street in the sentence? As this question may appear simple for human, it is extremely difficult for an algorithm. When the model processes the input, the attention mechanism looks at the other clues in the input and assign higher weights to better encode each word. As a result, the word “it” is associated with “animal” word.

Figure 13. The attention mechanism allows to encode the word “it” based on focusing on “The animal” words.

Attention was first introduced in NLP for machine translation tasks by Bahdanau, and Cho Bengio (2015). They proposed to utilize a context vector to align the source and target inputs. The context vector contains information from all hidden states from encoder cells and align them with the target output. Thus, the model can attend some elements of the source input and learn complex relationships between the source input and the target output. Shen et al (2018) proposed two novels attention mechanisms: (i): multi-dimensional attention that performs a feature-wise selection over the input sequence for a specific task and (ii): Directional Self-Attention that produces the context representations with temporal information encoded. Based on these proposed attentions, a
Directional Self-Attention Network (DiSAN) has been proposed for sentence encoding without recurrences or convolutions. Their experimental results showed that this model can achieve state-of-the-art results in various NLP tasks as they illustrated using various datasets for various tasks: the Stanford Natural Language Inference (SNLI) dataset, the Stanford Sentiment Treebank (SST), Multi-Genre Natural Language Inference (MultiNLI), Sentences Involving Compositional Knowledge (SICK), Customer Review, MPQA, and TREC question-type classification and Subjectivity (SUBJ) datasets. Vaswani et al. (2017) has introduced a multi-head attention mechanism yielding state-of-the-art results in machine translation. The transformers based on this attention mechanism are good at parallelization, increasing the model’s performance. Added to this, the Transformer-based models overcome the main shortcoming of previous deep learning models in assessing the very short answers that consist of few words.

In this work, we solve the reference matching problem in student answers by adding a contextual information to the student answer as a concatenation of its corresponding tutor question and problem description. For the shortness of answers containing less than 5 words, we propose an Attention-based Transformer that uses a multi-head attention mechanism that encodes the semantics of student and reference answers based on the most relevant context words. Our proposed model’s architecture is a Siamese network based on the Transformer’s encoder since we model the problem as a binary text classification task in which student responses are categorized as correct or incorrect.

**Related Prior Work**

Following the trend of transformers, several NLP researchers have explored their potential for the task of short textual similarity (STS), our focus in this dissertation work. For example, Tang et al.
(2018) proposed a shared sentence encoder to improve the multilingual semantic textual similarity (STS) in low resource language with insufficient labelling (e.g., Spanish, Arabic, Thai etc.). By exploiting the nature of a multilingual encoder, one sentence can have multiple representations for different target translation language which led to improved similarity results. Their proposed encoder STS model architecture consists of the following components: 1) word embedding, 2) masked multi self-attention and 3) Feed Forward Network. This Transformer-based model architecture is different from our proposed model in numerous ways. First, FastText (Bojanowski et al. 2017) embeddings were used only in their experiments. In our work, we used four different word embeddings: Glove, ELMo, FastText and Word2vec. Second, their attention mechanism is different from ours as it is based on an inter-sentence attention that follows the approach described in (Wang et al., 2016). Our attention mechanism is explained further in a later section. Finally, their proposed model was evaluated on a different task than ours. Our model is the first Encoder Transformer-based model applied for assessing the correctness of short student answers in the context of intelligent tutoring systems. In a continuous research effort, Yang et al. (2018) proposed an Encoder-based Network applied on the STS benchmark and SemEval 2017’s Community Question Answering (CQA) question similarity subtask. An Encoder-Transformer, as described in Vaswani et al. (2017), has been used to compute a sentence embedding of the input $u$ and the response embeddings $v$ which are passed through an additional fully connected layer to get the output $v'$. The final dot product between $u$ and $v'$ is computed to get the semantic score between the input and the response. The results of the conducted experiments for the STS Benchmark showed the competitiveness of the sentence encoding based models.

Transformers have been applied successfully in other NLP tasks. In machine translation, Transformer-based models achieved the state-of-the-art results. Tubay et al. (2018) have applied
a Transformer for a machine translation with multi-source Romance languages. It has been evaluated for the biomedical WMT 2018 task. The proposed Transformer consists of an encoder and decoder where the multi-head attention mechanism is the primary mechanism. The results of their experiments showed an improvement of over 6 BLEU points. In a continuous research effort, Bapna et al. (2018) trained significantly deeper Transformer and Bi-RNN encoders for machine translation. In their work, they used the latest version of the Transformer implementation from (Chen et al., 2018). One major challenge during experiments is the difficulty to train Transformer models when the encoder depth increases beyond 12 layers. To solve this problem, a transparent attention mechanism has been proposed. The results showed consistent gains in translation quality on both the WMT’14 (Eng → De) and WMT’15 (Cs→En) data. Ahmed et al. (2018) proposed a weighted Transformer with a modified attention layer. Unlike the classical Transformer, which weighs all heads equally (as we do in our work), the proposed attention mechanism allows for ascribing importance to different heads. This prioritizes their gradients and eases the optimization process. The experimental results showed that the proposed model improves the state-of-the-art performance by 0.5 BLEU points on the WMT 2014 English-to-German translation task and by 0.4 on the English-to-French translation task. In addition, the model converges 15-40% faster than the baseline network. Transformers have been applied successfully in the sentiment analysis task as well. Kant et al. (2018) have applied an Attention-based Transformer Network as described by Vaswani et al. (2017) for a multi-emotion sentiment classification task. The model has achieved a 0.69 F1 score on the SemEval Task, which is competitive with state-of-the-art models. The model has demonstrated high performance in a text classification task as well. Letarte et al. (2018) has proposed a self-attention network for text classification. One key difference between their model and Vaswani’s Transformer is that they only perform input-input attention with self-attention, as
they do not have sequences as output (the output is text classification). The experimental results showed a performance improvement of around 2% when using self-attention compared to a baseline without attention.

Motivated by these successes of applying the Transformer in multiple NLP tasks, we propose an Attention-based Transformer, as described next, for the student answers assessment task.

**Model Architecture**

Our proposed model consists of several important components: 1)- an embedding layer, 2)- a positional encoding layer, 3)- a Transformer layer, and 4)- a SoftMax classifier (see Figure 14). We consider an extended student answer as the first input consisting of the concatenation of the corresponding problem description, the previous tutor question (which accounts as prior context), and the student answer. The inputs to the embedding layers are tokenized.

**Embedding Layer**

Given a student answer $X$ and a reference answer $X'$, we tokenize them into a sequence of word tokens: $X = [w_1, \ldots, w_n]$ and $X' = [w'_1, \ldots, w'_m]$. Afterwards, each token is converted into a $d$-dimensional ($d=300,1024$) vector through the embedding layer. We considered the following four-word embeddings: Glove (Pennington et al., 2014), Word2vec (Mikolov et al., 2013), ELMo (Peters et al., 2018) and FastText (Bojanowski et al., 2017).
Figure 14. the Attention-based Transformer architecture. It consists of the following components: (i) an embedding layer, (ii) a Transformer Encoder, and (iii) a SoftMax layer.

**Positional Encoding**

Positional encoding is used to capture the order of the tokens in the input. Since the embedding layer captures the meaning of words and there is no recurrence and convolution in the proposed transformer network, the positional encoding is added to the input embeddings to inject the token order information. The positional encoding outputs have the same dimension $d_{model}$ as the embedding outputs so they can sum up.

The position encodings are calculated using the sine and cosine functions as in the following:
\[ PE_{\text{pos},2i} = \sin \left( \frac{\text{pos}}{10000^{2i/d_{\text{model}}}} \right) \]

\[ PE_{\text{pos},2i+1} = \cos \left( \frac{\text{pos}}{10000^{2i/d_{\text{model}}}} \right) \]

where \( \text{pos} \) is the position and \( i \) is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid.

For example, if we assume that the embedding has a dimensionality of 4, the actual position encoding would look as shown below:

![Position Encoding Example](image)

Figure 15. Position Encoding Example

The resulting sum vector of each embedding vector and its position encoding vector is imported into the multi-attention head mechanism.

**Transformer Layer**

The Transformer consists of a stack of identical encoder layers (see Figure 16). Each encoder is composed of two major components: 1) Multi-Head Attention mechanism and 2) Position-Wise Feed-Forward Network. What follows is a detailed explanation of each component.
Figure 16. Transformer architecture that consists of several identical encoders (left).

Encoder structure (right).

**Multi-Head Self-Attention Mechanism**

The multi-head attention mechanism, as depicted in Figure 17, consists of several attention layers running in parallel. This mechanism has been introduced by Google and uses multiple iterations of computation to capture relevant information. Added to this, this component improves the performance of the attention layer in two ways. First, it expands the model’s ability to focus on different positions. Second, it gives the attention layer multiple “representation subspaces”. This will be explained further in more details. The major advantage of self-attention is that it ignores the distance between words and computes directly dependency relationships. Thus, making it capable of learning the internal structure of a sentence.
Figure 17. Scaled dot product attention (left). Multi-Head Attention Structure (right).

An attention function consists of mapping a query and a set of key-value pairs to an output, where the query, keys, values and output are all vectors. Given a word vector \( x_1 \), calculating the self-attention vector \( z_1 \) consists of several steps.

- **The first step** consists of creating a Query vector, a Key vector, and a Value vector for each word embedding vector \( x_1 \). Three weight matrices are learned during the training process: \( W_q \), \( W_k \) and \( W_v \). Multiplying \( x_1 \) by \( W_q \) weight matrix results in creating the query \( q \) of the associated word. Similarly, we create the Key and Value vectors associated with each word, by multiplying \( x_1 \) by \( W_k \) and \( W_v \), respectively.

- **The second step** consists of calculating a score by taking the dot product of the Query vector \( q_1 \) and the Key vector \( k_1 \) associated with the word \( w_1 \) in the input. For the second word \( w_2 \) in the input, its score is calculating by multiplying \( q_1 \) and its Key vector \( k_2 \).

- **The third step** consists of dividing the scores by the square root of the dimension of the key vectors \( \sqrt{d_k} \). To normalize the scores, we pass them through a SoftMax operation.
The fourth step consists of multiplying each Value vector $v_1$ with the resulted SoftMax score. This helps to assign the most relevant words high scores. Then, we sum up the weighted values vectors to produce the output vector $z_1$ of the self-attention layer that corresponds to the word $w_1$ in the input. The resulting vector is passed to the Feed-Forward network.

As mentioned by Vaswani et al (2017), the self-attention calculation can be done using matrices instead of vectors. In this case, we can compute the multi-head attention as follows:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\text{head}_2,\ldots,\text{head}_n)W^0$$

where  

$$\text{head}_i = \text{Attention}(QW_i^Q,KW_i^K,VW_i^V)$$

$$\text{Attention}(Q,K,V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right)V$$

where the queries, keys and values are packed into matrices $Q,K and V$.

Since the multi-head attention component is based on multiple attention heads, we end up with multiple outputs from each attention head. And since the Feed Forward Network accepts one input, we concatenate the attention heads ($z_1,\ldots,z_n$) associated with each input and the multiply this concatenation with a learned weight matrix $W_0$. This produces the final outputs $Z$ and $Z'$ of the
multi-head attention component for the student answer input $X$ and the reference answer input $X'$, respectively.

**Position-Wise Feed-Forward Networks**

The resulting vector of the multi-head attention module $Z$ is passed through a fully connected forward network that computes linear transformations of the input. In this work, we consider two 1-dimension convolutions with kernel size $d_{\text{inner\_head}}$, a dropout to avoid overfitting and a normalization layer (see Table 13). The dimensionality of input and output is $d_{\text{model}}$.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Kernel size</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>$d_{\text{inner_head}}$</td>
<td>1</td>
</tr>
<tr>
<td>ReLU</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Convolution</td>
<td>$d_{\text{inner_head}}$</td>
<td>1</td>
</tr>
<tr>
<td>ReLU</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Dropout</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Layer</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Normalization</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 13. Feed Forward Network Architecture

Afterwards, we concatenate the outputs of the Feed-Forward Network $[v_1, v_2]$ and pass it to the final SoftMax layer to compute classification probabilities $p_1, p_2$ of belonging to the two classes: correct or incorrect.

**Experiments**

We test the performance of the proposed Attention-based Transformer using the previously introduced DT-Grade dataset. To this end, several experiments have been conducted using different parameters settings as explained next. To demonstrate the effectiveness of our proposed model, we compare the obtained results with previous results using previous applied deep learning models on the same dataset.
Data

In this work, we use the DT-Grade dataset as it has introduced previously. It consists of 900 instances of student answers extracted from logged tutorial interactions between 40 junior level college students and the state-of-the-art intelligent tutoring system DeepTutor. Each annotation example (See Figure 18) consists of the following attributes: (1) problem description, (2) tutor question, (3) student answer and (4) reference answers. In addition, the data includes the correctness class of each student answer. There are four classes: correct, correct but incomplete, incorrect, and contradictory. In this work, we consider two classes: correct and incorrect. The correct answers are those labeled as “correct” in the DT-Grade dataset. All the other instances are considered as belonging to the “incorrect” class.

Figure 18 Snapshot of the DT-Grade raw dataset in XML form. The main attributes are:(i) Problem Description, (ii)-Question(iii)Student Answer and (iv) Reference Answers
Experimental Setting
To evaluate the importance of the components of our attention-based transformer, we have conducted several experiments by varying the Transformer architecture and using different embedding approaches.

A first set of experiments have been conducted using Word2vec embeddings with 300 dimensions and different settings of the attention-based transformer. Based on the experiments conducted by Vaswani et al. (2017), we have tried various values of the number of attention heads \( (8, 10, 15, 32) \). This has been done to test the impact of increasing and/or decreasing the number of attention heads \( (n_{\text{head}}) \) on the performance of our model. We have, also, varied the depth of the transformer by experimenting with different number of the encoder layers. Other parameters have been modified as well such as the attention key dimension \( (d_k) \), the attention value dimension \( (d_v) \) and the number of the kernel size \( (d_{\text{inner,head}}) \) of the convolution layers in the Feed-Forward Network.

Another set of experiments has been conducted using Glove pre-trained embeddings with 300 dimensions. Following the same setup of the first set of experiments, we have used the same values of the same parameters: number of heads of attention, the attention key dimension, the attention value dimension and the number of layers.

As stated in several research works, ELMo embedding enhanced the performance of several deep learning models applied to various NLP tasks. For this reason, we have conducted another set of experiments using ELMo embeddings with 1,024 dimensions. We have followed the same setup of the previous sets of experiments and we have used the same values of the Transformer ‘s parameters. The only difference is the value of \( d_{\text{model}} \) that is set to 1024 to be consistent with the ELMo embedding dimension so we can sum up via the position encoding.
In a continuous effort to test the impact of the embedding approach on the performance of the proposed model, we conducted another set of experiments using FastText embeddings. We kept the same parameters settings of the previous experiments.

**Hyperparameters**

In all experiments, the model was trained with a Categorical Cross Entropy Loss function. For optimization, we used the Adam optimizer (Kingu and Ba, 2014) with a learning rate of 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The gradients are clipped to 0.5 to prevent exploding gradients. To avoid overfitting, we applied a dropout = 0.9 to the sums of the embeddings and the positional encodings of each layer of the Transformer. In all experiments, we trained our model for 1,000 epochs to obtain the results. An increasing number of epochs, particularly when using the ELMo embedding, showed an increase in the overall accuracy.

**Results & Discussion**

Table 14 shows the accuracy of different architectures of our model using the word2vec embedding. The highest accuracy of 59% was reached when using (15,8,16) heads of attention and (2,6,6) layers, respectively. This result outperforms Bi-GRU-Capsnet with word2vec embeddings (Ait Khayi & Rus, 2019) on the same dataset. The highest accuracy of 60% was reached when using 16 heads of attention and 6 layers of the encoder. It seems that increasing the number of heads of attention above 16 has led to a decrease in accuracy. This led us to our first observation: there are specific heads of attention that play an important role in the transformer and a specific
number of these heads of attention is sufficient to achieve good results. Thus, adding more heads of attention can be considered redundant for the transformer’s architecture. This performance is better than the performance obtained with the word2vec embeddings and outperforms the approach based on Bi-GRU-Capsnet with Glove embeddings.

Table 15 shows results for different architectures of our model using Glove embeddings. The highest accuracy of 60% was reached when using 16 heads of attention and 6 layers of the encoder. It seems that increasing the number of heads of attention above 16 has led to a decrease in accuracy. Similar to the results obtained using word2vec, we observe that there are specific heads of attention that play an important role in the Transformer and a specific number of these heads of attention is sufficient to achieve good results. Thus, adding more heads of attention can
be considered redundant for the Transformer’s architecture. This performance is better than the performance obtained with the word2vec embeddings.

Table 16 shows results for different architectures of our model using ELMo embeddings. We can observe a significant improvement in the overall accuracy in comparison with Glove and Word2vec embeddings. The highest accuracy of 71.5 was achieved when using 10 heads of attention and 2 layers only. This is a very competitive result in comparison with Bi-GRU-Capsnet and ELMo (72.5).

Table 16. Results of variations of the transformer architecture using Elmo embeddings.

<table>
<thead>
<tr>
<th>d_{model}</th>
<th>d_{inner_head}</th>
<th>n_{head}</th>
<th>d_k</th>
<th>d_v</th>
<th>layers</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>512</td>
<td>15</td>
<td>64</td>
<td>64</td>
<td>2</td>
<td>59</td>
</tr>
<tr>
<td>300</td>
<td>512</td>
<td>10</td>
<td>64</td>
<td>64</td>
<td>2</td>
<td>59</td>
</tr>
<tr>
<td><strong>300</strong></td>
<td><strong>2048</strong></td>
<td><strong>8</strong></td>
<td><strong>64</strong></td>
<td>64</td>
<td><strong>1</strong></td>
<td><strong>60</strong></td>
</tr>
<tr>
<td>300</td>
<td>2048</td>
<td>32</td>
<td>128</td>
<td>128</td>
<td>8</td>
<td>57</td>
</tr>
<tr>
<td>300</td>
<td>2048</td>
<td>8</td>
<td>64</td>
<td>64</td>
<td>6</td>
<td>57</td>
</tr>
<tr>
<td>300</td>
<td>512</td>
<td>10</td>
<td>64</td>
<td>64</td>
<td>1</td>
<td>58</td>
</tr>
<tr>
<td>300</td>
<td>4096</td>
<td>16</td>
<td>128</td>
<td>128</td>
<td>6</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 17. Results of variations of the transformer architecture using FastText embeddings

<table>
<thead>
<tr>
<th>d_{model}</th>
<th>d_{inner_head}</th>
<th>n_{head}</th>
<th>d_k</th>
<th>d_v</th>
<th>layers</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>512</td>
<td>15</td>
<td>64</td>
<td>64</td>
<td>2</td>
<td>61</td>
</tr>
<tr>
<td><strong>1024</strong></td>
<td><strong>512</strong></td>
<td><strong>10</strong></td>
<td><strong>64</strong></td>
<td><strong>64</strong></td>
<td>2</td>
<td><strong>71.5</strong></td>
</tr>
<tr>
<td>1024</td>
<td>2048</td>
<td>8</td>
<td>64</td>
<td>64</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>1024</td>
<td>2048</td>
<td>32</td>
<td>128</td>
<td>128</td>
<td>8</td>
<td>61</td>
</tr>
<tr>
<td>1024</td>
<td>2048</td>
<td>8</td>
<td>64</td>
<td>64</td>
<td>6</td>
<td>61</td>
</tr>
<tr>
<td>1024</td>
<td>512</td>
<td>10</td>
<td>64</td>
<td>64</td>
<td>1</td>
<td>67</td>
</tr>
</tbody>
</table>

Table 17 shows results of the transformer using FastText embeddings. The highest accuracy of 60% has been achieved when using 8 attention heads and 1 stack layer of encoders. This is the same accuracy obtained when using the transformer with Glove embeddings but with different values of the transformer’s parameters.
The depicted results in Table 18 show that the attention-based transformer has outperformed the Bi-GRU Capsnet when using the Glove embeddings: 60% versus 56.25% accuracy. It has outperformed also the Bi-GRU Capsnet when using the word2vec embeddings: 59% versus 56.25. The highest accuracy of 71.5 has been achieved with ELMo embeddings. The results show also that our proposed model displays a superior performance over the baseline models: Bi-GRU and LSTM models. An interesting finding in the conducted experiments (see some examples in Table 19) is that the proposed attention model handles the assessment of shorts answers with a small number of words less than 6 better than the recurrent networks: Bi-GRU and LSTM. This can be explained by the fact that the self-attention mechanism in our proposed model allows the selection of the most relevant words in the students answer and reference answer. Then, the assessment is computed based on those relevant words. As shown in Table 19, giving the following reference answer: “The ball is slowing down at a constant rate,”, the attention mechanism allows to focus on the most relevant part of this input:” slowing down”. This selected part has a similar semantic representation with the following student answer: “it is decreasing”. Thus, the Transformer is capable to assess this answer correctly. Similar observation has been made for additional short student answers with fewer words (see Table 19).

Table 18. Comparison between the Transformer and other deep learning models

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer (ELMo)</td>
<td>71.5</td>
</tr>
<tr>
<td>Bi-GRU-Capsnet (ELMo)</td>
<td>72.5</td>
</tr>
<tr>
<td><strong>Transformer (Glove)</strong></td>
<td><strong>60</strong></td>
</tr>
<tr>
<td>Bi-GRU-Capsnet (Glove)</td>
<td>56.25</td>
</tr>
<tr>
<td>Transformer (word2vec)</td>
<td>59</td>
</tr>
<tr>
<td>Bi-GRU-Capsnet (Word2vec)</td>
<td>56.25</td>
</tr>
<tr>
<td>Bi-GRU</td>
<td>56.25</td>
</tr>
<tr>
<td>LSTM</td>
<td>60</td>
</tr>
</tbody>
</table>
Table 19. Example of students and reference answers and their assessment using Bi-GRU and ELMo-Transformer

<table>
<thead>
<tr>
<th>Student answer</th>
<th>Reference answer</th>
<th>Ground Truth</th>
<th>Bi-GRU</th>
<th>Elmo Transformer</th>
</tr>
</thead>
<tbody>
<tr>
<td>second</td>
<td>When the mover doubles his force, the push is greater than friction resulting in a non-zero net force acting on the desk and so Newton's <strong>second law</strong> can be applied.</td>
<td>Correct</td>
<td>Incorrect</td>
<td>Correct</td>
</tr>
<tr>
<td>it is decreasing</td>
<td>The ball is <strong>slowing down</strong> at a constant rate.</td>
<td>Correct</td>
<td>Incorrect</td>
<td>Correct</td>
</tr>
<tr>
<td>They balance each other</td>
<td>Because the net force on the desk is zero, <strong>the mover's push balances the opposing force of friction.</strong></td>
<td>Correct</td>
<td>Incorrect</td>
<td>Correct</td>
</tr>
<tr>
<td>Equal to zero</td>
<td>Since the child is being raised straight upward at a constant speed, <strong>the net force on the child is zero</strong> and all the forces balance. That means that the tension in the rope balances the downward force of gravity.</td>
<td>Correct</td>
<td>Incorrect</td>
<td>Correct</td>
</tr>
</tbody>
</table>

**Conclusions**

Previous Bi-GRU Capsnet model suffers from its incapability to assess correctly the very short responses consisting of very few words less than 5 and pronouns referring to concepts in previous answers or the tutor question. In attempt to mitigate this issue, we proposed to add a contextual information to the student answer and an Attention-based Transformer to assess the correctness of student answers. Several experiments have computed data from tutorial interactions between students and the state-of-art of intelligent tutoring systems DeepTutor. This is the first time the Transformer Encoder (at the time of publication) has been applied to this task. We have chosen this approach to take full advantage of the self-attention mechanism in assessing accurately the
very short student answers. In addition, the Transformer has demonstrated a great effectiveness in numerous semantic textual tasks.

The proposed model is composed of several important components: i)- an embedding layer, ii)- a position encoding, iii)- a Transformer layer, and iv)- a SoftMax classifier. The experimental results on the DT-Grade dataset showed the high competitiveness of the proposed model in comparison with previous state-of-the-art approach. The highest accuracy of 71.5% was achieved using 10 heads of attention, 2 encoder layers and ELMo embeddings. This result is very close to the result achieved by Bi-GRU-Capsnet (72.5%) on the DT-Grade dataset.

In the future, we plan to further investigate other novel deep learning models that work well in low resource scenarios such ours, in attempt to enhance the current performance results on the DT-Grade dataset.
Chapter 5

Towards Assessing Open-Ended Students Answers Using Graph Convolutional Networks

Introduction

Automatic assessment of open-ended students answers or called by some researchers; automatic short answers grading (ASAG) is a fundamental research problem in natural language processing (NLP). This assessment consists of comparing semantically a student answer with an expert generated answer. A high similarity implies the correctness of the student answer.

Graph Neural Networks (GNN) were first proposed by Scarsell and colleagues (2009). It is an extension of Neural Networks for processing data represented in graph domains. In a graph, each node is naturally defined by its features and the related nodes. Recently, GNNs attracted a wide attention in the field of NLP because of its impressive performance, interpretability and the great expressive power of graphs to represent unstructured data such as text. In this context, Zayats and colleagues (2017) presented a novel approach for modelling threaded discussions on social media using a Graph-Structured Bidirectional LSTM that represents both hierarchal and temporal conversation structure. The experimental results showed the superior performance of the proposed model over the node-independent model for the popularity prediction task. In a continuous research efforts, Beck and colleagues (2018) proposed a model for Graph-To-Sequence learning that uses recent advances in encoder-decoder architectures. The model has been evaluated in two NLP tasks: generation from abstract meaning representations and machine translation. The experimental results showed the effectiveness of this proposed approach. Hamilton and colleagues (2017) presented GraphSAGE, a general inductive framework that leverages node feature information to efficiently generate node embeddings for previously unseen data. The performance
of the proposed GrapSAGE has been tested on the following tasks: (i)- classifying academic papers into different subjects using the Web Science citation dataset, (ii)- classifying Reddit posts to different communities, and (iii) classifying protein functions across various interaction graphs. The results reinforce the effectiveness of the proposed approach that outperformed all the baselines (e.g. random classifier, regression-based classifier, DeepWalk algorithm etc..) with significant margin.

Motivated by the successes of GNNs in numerous NLP tasks, we propose a Graph Convolutional Network (GCN), which is class of GNN, for student answers assessment task. First, we built a knowledge graph from the DT-Grade dataset. Then, we imported it to two layers of GCNs. Finally, we applied a SoftMax classifier.

**Related Work**

Joining the efforts of several researchers in generalizing Convolutional Neural Networks (CNN) to work on arbitrarily structured graphs, Kipf and colleagues (2017) introduced a simplified Graph Neural Network called Graph Convolutional Network (GCN) yielding the state-of-the-art results on multiple graph datasets. The application of GCNs in various NLP tasks led to very promising results. For example, Sahu and colleagues (2019) proposed a novel inter-sentence relation extraction model that builds a labelled edge GCN on a document-level graph. The graph is constructed with words as nodes and multiple intra- and inter-sentence dependencies between them as edges. Then fed to a GCN model to encode the graph structure and a multi-instance learning is utilized with bi-affine pairwise scoring to predict the relation of an entity pair. The experimental results showed that the model has achieved a comparable performance to state-of-the-art neural models on the inter-sentence relation extraction task. Working on the same task, Zhang and colleagues (2019) proposed a novel model for the relation extraction task. Their model consists of
the following components: 1) an instance encoder based on Convolutional Neural Networks (CNN) to encode the instance semantics into a vector, 2) a relational knowledge learning component that employs graph convolutional networks to learn explicit relational knowledge, and 3) a knowledge-aware attention component to select the most informative instance that matches the relevant relation. The experimental results showed that this model outperforms several baselines such as CNN. GCNs have been applied successfully as well for the semantic role labeling task that can be described as the task of discovering in texts who did what to whom. To this end, Marcheggiani and colleagues (2017) have proposed a model that consists of the following components: 1) word embeddings, 2) a Bi-LSTM encoder that takes as input the embedding representation of each word, 3) a syntax based GCN encoder that re-encodes the Bi-LSTM representation based on the predicted syntactic structure of the sentence, and 4) a classifier to predict the role associated with each word. The empirical results showed that this based GCN model has achieved the state-of-the-art results. GCNs have been explored successfully in text classification. For this purpose, Yao and colleagues (19) proposed to use GCNs for text classification. They built a single text graph for the whole corpus based on word co-occurrence and document word relations then learnt a text graph convolutional network for the corpus. The proposed model has been evaluated using multiple benchmarks. The experimental results showed that GCN outperforms several baselines such as Bi-Directional LSTM and LSTM. In this work, we don’t consider a heterogenous graph where nodes present words and documents. The nodes represent documents only as a concatenation between student answers and reference answers. Based on these successes of GCN in numerous NLP related tasks, we have explored their potential for assessing student natural language answers. To the best of our knowledge, this is the first attempt at using GCNs for this task. GCNs have demonstrated impressive results in the question
answering task as well which is related to the students answers assessment task. To this end, Cao and colleagues (2019) proposed a Bi-Directional Attention Graph Convolutional Network (BAG). The graph is built from documents with multi-level features where nodes are entities and edges are relationships between them. The graph is then imported into GCNs to learn relation-aware representation of nodes. Finally, for final prediction, a bi-directional attention is introduced between the graph and a query to derive the mutual information. The experimental results showed that BAG achieves the state-of-the-art results on the used dataset. In a continuous work on the same task, De Cao et al (2019) proposed to model the question answering problem as an inference problem on graphs built from documents. The nodes are entities and edges represent the relationship between different mentions such as within and cross-documents coreferences. The built graph is then fed into GCNs to perform multi-step reasoning. The experimental results show that this proposed model achieved the state-of-the-art results on a multi-document question answering dataset, WIKIHOP (Welbl et al., 2018).

In addition, GCNs have been explored successfully in other NLP tasks such as sentiment aspect classification (Zhang et al., 2019), abnormal text detection (Mishra et al., 2019; Li et al., 2019) emotion recognition (Deepanway et al., 2019).

**Model Architecture**

Our proposed method (see Figure 19) consists of building first a knowledge graph from the DT-Grade data. The built graph is imported into two GCN layers. Finally, we apply a classifier to predict the class of each text node.

**DT-Grade Graph**

We build a text graph from the DT-Grade dataset based on the citation relation approach (Kipf et al., 2017). We consider each document, whose content is the combination of the student answer
Figure 19: The model architecture consists of the following components: i) building a DTGraph, ii) feeding it to two GCN layers, iii) and finally applying a classifier and its corresponding reference answer, as a node which represents a specific question in Physics (e.g., What is Newton Law?). Thus, the classification of a pair of student answer and reference answer turns to a node classification task. The number of the nodes in the text graph is 900 which is the number of instances in the DT-Grade dataset. Formally given a graph \( G = (V, E) \) where \( V \) and \( E \) are sets of nodes and edges. The weight of the edge between two nodes is calculated using two methods: a TF-IDF method and an embedding based method. In the first one, we compute the term frequency-inverse document frequency (TF-IDF) between two text nodes. We add an edge between two nodes if the cosine similarity is above a threshold of 0.9. The second method is based on word2vec embeddings. First, word2vec is used to learn a vector representation for each word in the text representing each node. Then, we compute the Word Mover’s Distance (WMD) to measure the similarity between two texts representing two nodes in the graph. Texts that share many words should have smaller distances than texts with very dissimilar words. WMD has been
introduced to measure the distance between two text documents that takes into account the alignments between words. In this paper, we consider the text associated with each node as a short document. The WMD algorithm finds the values of an auxiliary ‘transport’ matrix $T$, such that $T_{ij}$ describes how much $d_i^a$ should be transported to $d_j^b$. The WMD learns $T$ to minimize:

$$D(x_i, x_j) = \min_{T \geq 0} \sum_{i,j=1}^n T_{ij} \|x_i - x_j\|_2^p$$

Subject to: $\sum_j T_{ij} = d_i^a$, $\sum_i T_{ij} = d_j^b$ $\forall$ $i,j$

where: $d_i^a$ and $d_j^b$ are the n-dimensional normalized bag-of-vectors for the two nodes’ texts, $x_i \in R^d$ is the embedding vector of the ith word and $p$ is usually set to 1 or 2.

The resulted graph is fed afterwards into a two-layer GCN, as explained next.

**Graph Convolutional Networks (GCN)**

GCN is a recent class of multilayer neural networks that operate on graphs (Duvenaud et al., 2015; Kipf et al., 2017). For every node in the graph, GCN encodes relevant information about its neighborhood as a real-valued feature vector. Formally given a graph $G = (V, E)$ where $V$ and $E$ are sets of nodes and edges. Every node is assumed to connect with itself, i.e., $(v, v) \in E$ for any $v$. Let $X \in R^{n \times m}$ be a matrix containing all $n$ nodes with their features, where $m$ is the dimension of the feature vectors, each row $x_v \in R$ is the feature vector for $v$. We consider $A$ an adjacency matrix of the graph $G$ and its degree matrix $D$ where $D_{ii} = \sum_j A_{ij}$. When using GCN with multiple layers, the information about larger neighbors is captured. Following the recommendation of Kipf
et al (2017) that multiple layers yield in better performance, we consider two layers of GCN. The new $k$-dimensional node feature matrix of layer $L^{(j+1)}$ is computed as following:

$$L^{(j+1)} = \rho(\tilde{A}L^{(j)}W_j)$$

Where $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the normalized symmetric adjacency matrix and $W_j$ is a weight matrix and $\rho$ is an activation matrix and $L^{(0)}=X$.

In this thesis work, we are interested in implementing a convolution operation in the spectral domain with a nonlinear trainable filter that maps the node features in a new space (Bruna et al., 2013; Henaff et al., 2015). To this end, we consider the following filters:

- **Local pooling filter**: Kipf and colleagues (2017) proposed spectral convolutions on graph defined as the multiplication of a scalar for every node with a filter in the Fourier domain.

- **Chebyshev polynomial filter**: Chebyshev polynomials are exploited to implement fast localized filters in a GCN as described in (Defferrard et al., 2016). The use of this filter allows avoiding to eigen-decompose the Laplacian by approximating the filter convolution.

- **Auto-Regressive Moving Average (ARMA) filter**: Bianchi and colleagues (2019) have proposed a Graph Neural Network that implements convolutional layers based on an ARMA filter that provides a more flexible response. The experimental results showed that ARMA outperforms the polynomial filters.
SoftMax Classifier

The output of the second GCN layer is fed into a SoftMax layer as follows:

\[ Z = \text{softmax}(\tilde{A}\text{ReLU}(\tilde{A}XW_0)W_f) \]

Where \( \tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \) is the normalized symmetric adjacency matrix, \( W_f, W_0 \) are weight parameters and \( \text{softmax}(x_i) = \frac{\exp(x_i)}{\sum \exp(x_i)} \).

\( \tilde{A}XW_0 \) contains the first layer document embeddings and \( (\text{ReLU}(\tilde{A}XW_0)W_f) \) contains the second layer document embeddings.

Experiments

Several empirical experiments have been conducted to test the performance of the proposed GCN based model on the DT-Grade data as explained in a previous section. For this purpose, the model has been tested with different parameters settings and different filters. Then has been compared with previous methods.

Experimental Setting

To evaluate our proposed method, we have conducted numerous experiments with different parameters settings. To this end, we train and evaluate a two-layer GCN, as described previously, using the DT-Grade dataset. In all experiments, we train our model for a maximum of 1000 epochs (training iterations) using Categorical Cross Entropy Loss function and Adam optimizer (Kingma & Ba., 2014) with a learning rate of 0.01. And we stop the training if the validation loss does not
decrease for 100 consecutive epochs, as suggested in work (Kipf et al., 2017). To avoid overfitting, we apply a dropout = 0.5. For the graph convolution layer, we use a hidden layer size of 16 units with an L2 regularization and ReLU activation. We select randomly 600 instances to evaluate the training accuracy, 100 instances to evaluate the validation accuracy and 200 instances to assess the testing accuracy.

In first set of experiments, we have used the TF-IDF approach to compute the weight of the DT-Grade graph edges. Then we repeated the experiment with the following filters as described in the previous section: 1- local pool filter which is considered as a baseline filter for Graph Convolutional Networks, 2- Chebyshev polynomial filter with two values of the maximum polynomial degree (2,3) and 3 – ARMA filter. We report the accuracy of the model for each filter. We repeat the same set of experiments using the embedding based approach to extract features.

In a second set of experiments, we have used word2vec embedding with 300 dimension and WMD to compute the weight of the edges. Then, we report the accuracy of the model using three different filters as described previously. We repeated the same set of experiments with different methods of features extraction.

Results & Discussion

Table 20 summarizes the performance results of using GCN with different parameters settings. Several observations can be made of these obtained results. First, the use of TF-IDF method to compute the weights between the edges outperformed the word2vec based method in all experiments. This result is in line with the fact that TF-IDF method captures more informative information between two texts presented by two nodes. The highest accuracy obtained within TF-IDF was 73 % versus 70% of the word2vec method. Second, the experimental results show
that applying ARMA filter has outperformed the other polynomial filters. A highest accuracy of 73% has been achieved. This is consistent with the work of Bianchi and colleagues (2019). In another set of experiments, we varied the maximum polynomial degree using two values: 2 and 3. The depicted results in Table 20 show that when using the TF-IDF and the value of 2, a higher accuracy of 72% has been achieved versus 70% for the value =3. These two values have been selected from the literature.

Table 20. Performance of GCN with different filters

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN (TF-IDF+localpool filter)</td>
<td>68</td>
</tr>
<tr>
<td>GCN (TF-IDF+Chebyshev filter K=2)</td>
<td>72</td>
</tr>
<tr>
<td>GCN (TF-IDF+Chebyshev filter K=3)</td>
<td>70</td>
</tr>
<tr>
<td><strong>GCN (TF-IDF+ARMA filter)</strong></td>
<td><strong>73</strong></td>
</tr>
<tr>
<td>GCN (word2vec+WMD+localpool filter)</td>
<td>62.5</td>
</tr>
<tr>
<td>GCN (word2vec+WMD+chebyshev filter K=2)</td>
<td>69.5</td>
</tr>
<tr>
<td>GCN (word2vec+WMD+chebyshev filter K=3)</td>
<td>70</td>
</tr>
<tr>
<td>GCN (word2vec+WMD+ARMA filter)</td>
<td>70</td>
</tr>
</tbody>
</table>

To test the impact of the number of units in the hidden layer on the performance of the models, we have carried another set of experiments by varying the number of the units in the hidden layer of the GCN when using the TF-IDF approach and the ARMA filter. The highest accuracy in the all experiments of 73% has been achieved when using 20 units in the hidden layer. The lowest accuracy of 69% has been achieved when using 68 units. It seems that increasing the number of units hurts the performance of the GCN. The results are depicted in the following table.

Table 21. Performance of GCN with different number of units

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN (TF-IDF+ARMA filter+ 16 units)</td>
<td>72.5</td>
</tr>
<tr>
<td><strong>GCN (TF-IDF+ARMA filter+ 20 units)</strong></td>
<td><strong>73</strong></td>
</tr>
<tr>
<td>GCN (TF-IDF+ARMA filter+ 30 units)</td>
<td>72</td>
</tr>
<tr>
<td>GCN (TF-IDF+ARMA filter+ 40 units)</td>
<td>71</td>
</tr>
<tr>
<td>GCN (TF-IDF+ARMA filter+ 50 units)</td>
<td>69</td>
</tr>
</tbody>
</table>
Table 22 summarizes a performance comparison between our proposed model and other deep learning models that have been applied on the DT-Grade dataset. The results show that baselines such as LSTM and Bi-GRU perform less than the other models. The proposed GCN model surpasses the previous deep learning models yielding the state-of-the-art results on the DT-Grade dataset at the time of publication.

Table 22. Comparison between GCN and other models for the DT-Grade dataset

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN (TF-IDF+ARMA filter+20 units)</td>
<td>73</td>
</tr>
<tr>
<td>LSTM</td>
<td>60</td>
</tr>
<tr>
<td>Bi-GRU</td>
<td>56.25</td>
</tr>
<tr>
<td>Transformer Encoder +ELMo</td>
<td>71</td>
</tr>
<tr>
<td>Bi-GRU Capsnet+ELMo</td>
<td>72.5</td>
</tr>
</tbody>
</table>

Conclusions

Motivated by the good results of applying Graph Convolutional Networks (GCN) in various NLP tasks, we propose to use a GCN based model to assess the correctness of student answers in conversational intelligent tutoring systems. It should be noted that is the first time such model is applied for this task. The results demonstrated the effectiveness of the proposed model by yielding the state of the-art results on the DT-Grade dataset (at the time of the publication). A highest accuracy of 73 % has been achieved when using the TF-IDF and the ARMA filter.

Despite the good results obtained by the proposed GCN model on our DT-Grade dataset, they still less promising compared to the results obtained for the large NLP datasets. This is due to the small size of our annotated data as collecting and labelling data for this domain is very expensive and tedious. To overcome this limitation, we plan in the future to apply a transfer learning approach based on the pretraining-finetuning paradigm.
Chapter 6  

Towards Improving Open Student Answer Assessment using Pretrained Transformers  

Introduction  

In the recent past, researchers have made significant progress in solving the open student answer assessment task while accounting for context and other knowledge sources using deep learning methods (Khayi et al., 2019, Khayi et al., 2020, Maharjan et al., 2018, Gong et al., 2019). The success of deep learning methods depends on the availability of large amount of high-quality labeled data. In many cases, including ours, the size of available data is small. An option to alleviate this limitation is using transfer learning models, the focus of our work presented here, to incorporate knowledge from other sources.  

The main idea behind transfer learning is to pretrain a model on large amounts of unlabeled data to obtain a powerful language model which can then be specialized for solving specific NLP downstream tasks by adding new layers and training them on the target data. These pretrained language models have been used recently to obtain state-of-the-art results in many NLP tasks (Devlin et al., 2019; Yang et al., 2019; Dong et al., 2019; Liu et al., 2019; Lan et al., 2019). Motivated by these successes, we explore the potential of finetuning several pretrained transformers on the student answers assessment downstream task. We experimented with such pretrained transformers on the DT-Grade dataset (Banjade et al. 2016) which contains 900 instances categorized in four classes: correct (367 instances), incorrect (238 instances), correct but incomplete (210 instances), and contradictory (84 instances). To overcome the problem of class size imbalance in the dataset and given its relatively small size, we considered a binary
classification where all instances in the incorrect, correct but incomplete, and contradictory categories are deemed as incorrect.

**Related Work**

The student answer assessment task has attracted broad attention recently. Several researchers have explored the potential of pretrained transformers as they led to state-of-the-art results in numerous NLP tasks. For example, Camus and colleagues (2020) experimented with fine tuning multiple pretrained transformers for the automatic short answer grading (ASAG) downstream task, which is related to our task, on the SemEval-2013 dataset. They also investigated the impact of transfer learning from the Multi-genre Natural Language Inference (MNLI) dataset to SemEval-2013 dataset on performance and generalization. The experimental results showed a significant gain of 15% improvement in performance score. The results also showed that distilled versions of the pretrained models with reduced parameters led to a slight decrease in the performance score but still feasible for the ASAG task.

Working on the same task using our DT-Grade dataset, Candor (2020) finetuned BERT on the ASAG downstream task. The model has been evaluated using Cohen’s Kappa as a measure of inter-rater reliability between the automated system and the human rater. The Experimental results showed that pretrained models such as BERT can help achieve more consistent human ratings. In their research efforts to improve the performance results of the ASAG task, Sung and colleagues (2019) proposed new ways to enhance the performance of BERT. To this end, they proposed to pretrain BERT on domain specific data such as textbooks and use labeled automatic short answer grading data to enhance the language model. Then, they finetuned the pretrained BERT model on the downstream task by considering two inputs: the student answer and the reference answer. The
experimental results showed that fine tuning BERT using the enhanced pretrained model achieves superior performance on the ASAG downstream task.

In this chapter, we explore for the first time the potential of pretrained transformers such as T5 and XLNET models and others for the open student answer assessment task.

**Methods**

**BERT** (Devlin et al., 2019): pretrains deep bidirectional representations from unlabeled text by jointly conditioning on both left and right contexts in all layers. The model is trained on the Book Corpus (Zhu et al., 2015) and English Wikipedia. The pair of sentences (student answer/A and reference answer/B) is packed into a single sequence and separated with a special token ([SEP]) and a classification token [CLS] at the beginning. An additional learned embedding is added to every token indicating whether it belongs to sentence A or sentence B. The resulted embedding $H$ of the [CLS] token is then fed into a SoftMax layer that predicts the probability of classification label $c$.

**T5** (Raffel et al., 2019): transforms all NLP tasks into a text-to-text format where the inputs and outputs are text strings. The model was pre-trained on the Colossal Clean Crawled Corpus (C4). The pre-training objective of T5 is similar to BERT’s with a small modification which is utilizing a Masked Language Model that masks 15% of the input tokens and replaces them with multiple mask key words instead of a unique one as in the case of BERT. Then, the model is trained to recover the masked tokens. T5 model’s architecture is based on both the encoder and decoder of the transformer (Vaswani et al., 2017).

**RoBERTa** (Liu et al., 2019): retrains BERT with an improved training methodology which involves 1,000% more data and computation power. RoBERTa has a different encoding
mechanism from BERT. That is, the student answer and reference answer are separated with two [SEP] tokens and a [CLS] token is added at the beginning.

**XLNet** (Yang et al., 2019): uses a permutation-based language modeling objective to capture bidirectional contexts while retaining the benefits of autoregressive (AR) models. Permutation language models are trained to predict one token given preceding context in some random order. Unlike the other previous pretrained models, the architecture of XLNet is based on the XL-Transformer (Dai et al., 2019). Similar to the finetuning used for BERT, we concatenate the student answer and reference answers separated with a [SEP] token. The [CLS] is added at the end.

**DistilBERT** (Sanh et al., 2019): uses a technique called distillation which approximates the large model of BERT with a smaller one. DistilBERT is distilled on very large batches by leveraging gradient accumulation using dynamic masking and without the next sentence prediction objective. The experiments have demonstrated a high impact of this reduction on computation efficiency. The encoding mechanism is similar to BERT.

**ALBERT** (Lan et al., 2019): has the same architecture as BERT. It implements two design changes that yields a model with 12M parameters, and 89% parameter reduction compared to the BERT model. This results in an efficiency improvement versus a minor performance degradation. The encoding mechanism of the student answer and reference answer is similar to the one we presented earlier for BERT.

**Experiments and Results**

We conducted experiments with the above-described methods using the DT-Grade dataset (Banjade et al., 2016) that was created by extracting student responses from logged tutorials interactions between 36 junior level college students and a state of the art conversational ITS.
We performed all our experiments using a Tesla K80 GPU and a total of 12 GB of RAM. All the models were implemented using the HuggingFace’s library (Wolf at al., 2019). We used the base versions of the pretrained transformers that are trained for 4 epochs. The Adam optimizer (Kingma et al., 2014) with a learning rate of 3e-5 was used and the gradients were clipped to 1.0 to prevent exploding gradients. We evaluated our models using the Sparse Categorical Cross-entropy loss and the Sparse Categorical accuracy. About 80% of data was used for training and 20% for testing. Each experiment was repeated 100 times with increased random seeds in an attempt to increase the models’ performance (Dodge et al., 2020). We report the best performance results across the 500 conducted experiments.

Table 23 shows performance results of finetuning the pretrained transformers on the DT-Grade dataset. As shown in the table, all the pretrained models outperform the previous methods with a significant margin. The T5 transformer has achieved the highest performance with an accuracy of 0.88 and an F1 score of 0.88. The results also showed that the distillation of BERT parameters is feasible for the student answers assessment task. ALBERT and DistilBERT have performed less than other pretrained transformers with an accuracy of 0.80 and an F1 score of 0.80. But still, it is a very good result in comparison with previous methods such as Bi-GRU-Capsnet (Ait Khayi et al., 2019), an attention-based transformer (Ait Khayi et al., 2020), and a Graph Convolutional Network (Ait Khayi et al., 2020). Another observation can be made from the obtained results is that BERT outperforms XLNET which works better for longer sequences, which is not the case of our student and reference answer which are relatively short.
Table 23: Performance results of the pretrained models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>0.86(+0.13)</td>
<td>0.86</td>
</tr>
<tr>
<td>RoBERTa</td>
<td>0.87(+0.14)</td>
<td>0.87</td>
</tr>
<tr>
<td>T5</td>
<td><strong>0.88(+0.15)</strong></td>
<td><strong>0.88</strong></td>
</tr>
<tr>
<td>XLNET</td>
<td>0.84(+0.11)</td>
<td>0.84</td>
</tr>
<tr>
<td>ALBERT</td>
<td>0.80(+0.7)</td>
<td>0.80</td>
</tr>
<tr>
<td>DistilBERT</td>
<td>0.80(+0.7)</td>
<td>0.80</td>
</tr>
<tr>
<td>Graph Convolutional Network</td>
<td>0.73</td>
<td>0.73</td>
</tr>
<tr>
<td>Bi-GRU-Capsnet+ELMo</td>
<td>0.72</td>
<td>0.70</td>
</tr>
<tr>
<td>Transformer+ELMo</td>
<td>0.71</td>
<td>0.70</td>
</tr>
<tr>
<td>LSTM+Glove</td>
<td>0.60</td>
<td>0.60</td>
</tr>
</tbody>
</table>

Overall, the experiments have demonstrated that these pretrained transformers assess correctly the very short answers in comparison with previous methods. For example, RoBERTa handles the assessment of very short student answers concatenated with reference answers with a small number of words (average of 10.5 words) better than the Bi-GRU-Capsnet network.

During the experiments, we investigated whether the learning rate and the sequence length parameters have an impact on the performance results. The experimental results showed that the smaller the learning rate the better the performance results. The value of 3e-5 has led to the best results versus the values of 4e-5 and 5e-5. The results also showed that the longer the length of the input sequence the better the performance results.

**Conclusions**

Several research studies have demonstrated the effectiveness of the transfer learning pretraining-finetuning paradigm for low resource scenarios in NLP as it is the case for the open student answer assessment task. Motivated by these successes with small datasets, we explored the potential of several pretrained transformers on the student answers assessment downstream task. To this end, we finetuned T5, XLNET, BERT, DistilBERT, ALBERT and RoBERTa transformers on the DT-
Grade dataset for the first time. The experimental results showed the effectiveness of these pretrained transformers that surpassed all the previous methods with a significant margin. The T5 transformer has achieved the highest performance with an accuracy of 0.88 and an F1 score of 0.88. This is a new state of the art on the DT-Grade dataset.

In the future, we plan to find better strategies to fine tune and pretrain these transformers on domain related data in order to improve the assessment results.
Chapter 7

Discourse based Automated Essay Scoring using XLNET Model

Introduction

As an extension of the short student answers assessment, we tackle the Automated Essay Scoring (AES) task, which is an important educational application in Natural Language Processing. It consists of evaluating and grading the quality of written natural language essays using machine learning. Most of the research work done in the AES task is based on a holistic approach, which summarizes the quality of an essay with a single score (Page, 1996; Phandi et al., 2015; Zesch et al., 2015; Taghipour et al., 2016; Dong et al., 2016, 2017; Zhang et al., 2018, 2020). This approach has been criticized of its inability to provide constructive feedback for the learner about which aspects of the essays need improvement. To overcome this drawback, several researchers in AES started to score a particular dimension of the essay quality such as sentence fluency (Chae et al., 2009), organization (Persing et al., 2010; Taghipour et al., 2017; Mathias et al., 2018; Song et al. 2020), sentence clarity (Persing et al., 2013; Ke et al. 2019), prompt adherence (Persing et al., 2014), argument strength (Persing et al., 2015; Taghipour et al., 2017), style (Mathias et al., 2018) and narrative quality (Somasundaran et al. 2018).

However, little attention has been paid to score the discourse aspect (i.e., structure) of the essay regardless its importance. Two types of discourse have been discussed in the literature: coherence and cohesion. A coherent essay contains related parts with strongly connected words. For example: “I was born in Glasgow. Glasgow is the largest city in Scotland”. Whereas an incoherent text contains unrelated parts. For example: “I was born in Glasgow. It is very nice in Scotland”. Cohesion refers to the presence or absence of linguistic cues in the text that allow the reader to make connections between the ideas in the text. Examples of these cues include
conjunctions such as discourse indicators (DIs) (e.g., “because” and “for example”), coreference (e.g., “he” and “they”), substitution, ellipsis, etc.

In general, the AES based models can be divided to two streams: features engineering based models and deep learning-based models that extract effective semantic features. The first consists of predicting the score of an essay using handcrafted features (e.g., spelling errors, length of essay etc.) and a simple regression model (Amorim et al., 2018; Litman et al., 2018). Although this approach has the interpretability and explainability advantages, the features extraction process is tedious and expensive to achieve high scoring accuracy. To alleviate this drawback, researchers have applied extensively deep models to obviate the need for the features engineering process (Yang et al., 2020; Mayfield et al., 2020; Rodriguez et al., 2019; Hirao et al., 2020; Nadeem et al., 2019; Farag et al., 2018). For instance, these two approaches can be considered complimentary since the handcrafted features can capture features that the neural network cannot extract from the text and vice-versa. To get the benefit of both approaches, several researchers recently proposed a hybrid approach which consists of incorporating expert features into deep learning models (Uto et al., 2020; Liu et al., 2019; Ridley et al., 2020). The experimental results demonstrated the effectiveness of this hybrid approach outperforming the traditional AES approaches with a significant margin.

Motivated by the successes of the hybrid approach in the AES task, we propose a hybrid model to evaluate the discourse aspect of an essay. Our model is composed of important components. Giving a written essay, we apply the pretrained XLNET based model to generate a distributed representation of the essay. Next, we concatenate this representation with several handcrafted discourse features. First, we select some novel discourse features that have not been explored in the AES task such as the lexical chains (Morris et al., 1991). Then we select another
set of discourse handcrafted features from the language analysis Coh-Metrix tool (McNamara et al., 2014) that do not correlate with the first set. Finally, we apply a linear function to predict the final score. To interpret the generated embedding features from XLNET, we computed their correlation with the Coh-Metrix based features.

**Related Work**

To take the benefit of the AES featuring engineering approach and the AES neural approach, several researchers recently proposed a hybrid approach that integrates both approaches. For example, Dasgupta and colleagues (2018) highlighted the limitations of current deep neural networks such as LSTM and CNN in identifying the interconnection between the different factors involved in assessing the quality of a text. To overcome this drawback and enhance the performance of the AES task, the authors proposed a deep neural network AES with an additional recurrent network that processes a sequence of several handcrafted enhanced features such as, lexical diversity, informativeness, cohesion and well-formedness. The experimental results showed that this hybrid approach has achieved the state-of-the-art result at the time of the publication. Uto and colleagues (2020) criticized the increased complexity of this previous framework because of the applied RNN on the handcrafted features which can negatively affects the training time. As a remediation of this raised issue, the authors proposed to apply a DNN on the essay to generate a distributed representation, then concatenate it with a handcrafted features-based vector (e.g., readability features, lexical features, syntactic features etc.). And finally, feed the merged vector to a linear layer to predict the final score. The authors proposed two types of DNN: 1- a recurrent based model such as LSTM and 2- a transformer-based model such as BERT. This approach can be applied on other DNN-AES models easily without increasing the model complexity and it improves the performance prediction. Adopting the same approach, Liu and
colleagues (2019) proposed Two-Stage Learning Framework (TSLF) which integrate both encoded features using DNN and handcrafted features. In the first stage, the authors proposed an LSTM based model to compute three different scores: 1- semantic score, 2- coherence score and 3- prompt-relevant score. In the second stage, the three scores are concatenated with handcrafted features (e.g. grammar errors, essay length in words and characters, vocabulary size etc.) and then fed to a boosting tree model to predict the overall score. The experimental results demonstrate the effectiveness and robustness of the TSLF framework which outperform many strong baselines such as CNN and LSTM on the five-eight prompts of the ASAP dataset. Ridley, and colleagues (2020) highlighted the problem of cross-prompt AES in the scenario where there are no labelled target-prompt essays available for training. To alleviate this issue, the authors proposed a neural network combined with traditional linguistic features, avoiding the need for pseudo-labelling, the need for abundant unlabeled target-prompt essays, and the need for suitable distribution of quality in the target-prompt essays. This proposed approach allows to avoid overfitting to the non-target-prompt essays. The experimental results demonstrated the effectiveness of the proposed method yielding the state-of-the-art result at the time of publication.

To get the benefits of the hybrid approach as stated previously in this chapter, we propose to evaluate the discourse aspect of an essay using a hybrid model. First, we extract the discourse embeddings of an essay using the pretrained XLNET model which suits better the long sequences (Yang et al., 2019). Second, we concatenate these generated embeddings with handcrafted discourse features derived from the lexical chains, the Coh-Metrix tool, and others. Finally, we apply a linear function to predict the overall score.
Proposed Model

Our proposed hybrid model (see Figure 20) consists of extracting an embedding based representation of an essay using the pretrained XLNET based model. Then we concatenate the generated embeddings vector with discourse handcrafted features. Finally, we apply a linear function to predict the final score.

![Model Architecture](image)

**Figure 20. Model Architecture**

XLNET Pretrained Model

The main reason of choosing XLNET (Yang et al., 2019) is that it includes segments of recurrence, introduced in Transformer-XL (Yang et al., 2019), allowing it to digest effectively longer texts. XLNet is a generalized autoregressive pretraining method that enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order.

Generally, XLNET has two parameter intensive settings:
- XLNETBASE: 12 layers, 768 hidden dimensions and 12 attention heads (in transformer) with the total number of 110 M parameters.

- XLNETLARGE: 24 layers, 1024 hidden dimensions and 16 attention heads (in transformer) with the total number of parameters, 340M.

**Two Stream Self Attention**

The main feature of the XLNET model is that it is based on the permutation language modeling that predicts the t-th word given the t-1 previous words as a context. The objective function of the permutation language modeling can be expressed as follows:

\[
\max_\theta E_{Z \sim Z_T} \left[ \sum_{t=1}^T \log p_\theta(x_{z_t} | X_{z < t}) \right]
\]

Where:

- \( Z \): set of all the possible permutations of the length of the T-index sequences \([1,..,T]\)
- \( p_\theta \): likelihood function
- \( x_{z_t} \): the t-th token in the factorization order
- \( X_{z < t} \): the tokens before the t-th token

However, there are two requirements that the transformer can’t do:

1. The prediction of the t-th token requires the information about its position in the original sequence and not its content. For instance, the transformer embeds the position of the token in its embedding and can’t separate this information.
2. The prediction of the t-th token requires encoding all the t-1 previous tokens as a content (semantics and syntactic).

XLNET resolves these issues by considering a two-attention mechanism:

- **Query Stream**: has access to the contextual information \( X_z(t) \) and the position \( z(t) \), but not the content \( X_c(t) \). This is initialized with a random weight.

- **Content Stream**: which is the standard self-attention mechanism in a transformer. It includes both contextual information of previous tokens \( X_A(t) \) and content at \( z(t) \). This is initialized with the corresponding word embeddings.

To understand the intuition behind the two-stream attention, we can think of XLNET replacing the [MASK] in BERT with query representation learned by query attention stream.

**Features Extraction**

Given an input essay of N tokens \([t_1, t_2, \ldots, t_N]\), each token is transformed to its embedding and passed to the base version of XLNET. Then we collect the output of the [CLS] token which is a vector \( H \) of 768 dimension and used as a text representation of the essay. Then, we concatenate the resulting \( H \) vector with a set of discourse handcrafted features.

**Discourse features**

They encode the discourse structure of an essay and have been derived from lexical chains. To be noted, this is the first time the lexical chains are explored in the AES task. Lexical chains represent sequences of related words semantically in a text. They have been used as an indicator of text cohesion (Morris et al., 1991). Intuitively, an essay that contains many lexical chains, especially ones where the beginning and end of the chain cover a large span of the essay, tend to be more
cohesive (Somasundaran et al., 2014). In this work we consider the following lexical chains features:

- Average chain size
- Number of large chains
- Percentage of large chains
- Number of varied chains
- Percentage of large, varied chains
- Number of large, varied chains

Additionally, we consider grammatical errors as a discourse measure (Burstein et al., 2013). This feature addresses errors in grammar that could interfere with a reader’s ability to construct meaning. We also consider the word unigrams, bigrams, and trigrams as they encode discourse information about essays (Ke et al., 2019). For instance, the bigram “people is” suggests ungrammaticality; the use of discourse connectives (e.g., “moreover”, “however”). The key advantage of using n-grams as features is that they are language-independent.

**Coh-Metrix features**

Coh-Metrix is a language analysis tool that assess texts via cohesion, coherence, and readability. It provides 110 metrics that are classified into 11 groups:

1. *Descriptive*: used to check the patterns in the text such as number of paragraphs, sentences, and words.

2. *Text Easability Principal Component Scores*: provide a clear picture about the text ease that emerge from the linguistic characteristics of the text. They are also aligned with theories of text and discourse comprehension.
3. *Referential Cohesion,* which assesses the number of cohesion relations that a human reader could do based on the propositions and sentences of the text.

4. *Latent Semantic Analysis,* which measure the semantic overlap between sentences and paragraphs. The scores range from 0 (low cohesion) to 1 (high cohesion).

5. *Lexical Diversity,* which measures the type of token ratios to deduce high cohesion

6. *Connectives,* which counts the incidence of connectives in the text.

7. *Situation Model,* which has been used in discourse processing to refer to the level of mental representation for a text that involves much more than the explicit words.

8. *Syntactic Complexity,* which syntactically analyzes the sentence and assesses the word density.

9. *Syntactic Pattern Density,* which assesses the incidence of different types of patterns in the texts.

10. *Word Information,* which shows the word type density in the text.

11. *Readability,* which assesses the text readability with formulae such as Flesch Reading Ease and Flesch-Kincaid Grade Level (Graesser et al., 2005).

**Linear Layer**

In this work, we proceed as a regression task. We use the following scoring function to map the essay representation $H$ to a scalar value by applying ReLU activation function.

$$ y = ReLU (w \cdot H + b) $$

where $w$ is the weight vector, $b$ is the bias and $y$ is the computed score.
Experiments and Results

To evaluate the performance and effectiveness of our proposed model, we have conducted several experiments using the Automated Student Assessment Prize (ASAP) dataset as described next.

ASAP dataset

The Automated Student Assessment Prize dataset contains eight prompts with different topics, including narrative essays, response essays and argumentative essays. In total, there are nearly 13000 essays in the dataset. Each of the essays was written by high school student belonging to classes 7 to 10. Each essay is assigned a score given by the instructors. The following table displays some statistics of the dataset.

Table 24. Statistics of the ASAP dataset; Range means the score range.

<table>
<thead>
<tr>
<th>Prompt</th>
<th># of Essays</th>
<th>Avg. Len</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1783</td>
<td>350</td>
<td>2-12</td>
</tr>
<tr>
<td>2</td>
<td>1800</td>
<td>350</td>
<td>1-6</td>
</tr>
<tr>
<td>3</td>
<td>1726</td>
<td>150</td>
<td>0-3</td>
</tr>
<tr>
<td>4</td>
<td>1772</td>
<td>150</td>
<td>0-3</td>
</tr>
<tr>
<td>5</td>
<td>1805</td>
<td>150</td>
<td>0-4</td>
</tr>
<tr>
<td>6</td>
<td>1800</td>
<td>150</td>
<td>0-4</td>
</tr>
<tr>
<td>7</td>
<td>1569</td>
<td>250</td>
<td>0-30</td>
</tr>
<tr>
<td>8</td>
<td>723</td>
<td>650</td>
<td>0-60</td>
</tr>
</tbody>
</table>

Experimental Settings

We performed all our experiments using a Tesla K80 GPU and a total of 12 GB of RAM. The model was implemented using the HuggingFace’s library (Wolf at al., 2019). We used the base version of the pretrained XLNET that is trained for 4 epochs. The maximum sequence length of XLNET is changed per prompt. The Adam optimizer (Kingma et al., 2014) with a learning rate of 1e-5 was used and the gradients were clipped to 1.0 to prevent exploding gradients. We evaluated
our model using the Mean Squared Error (MSE) loss. About 80% of data was used for training and 20% for testing. Each experiment was repeated several times and we selected the best model.

Before running the main experiment, we run the Coh-Metrix tool on the ASAP data to extract 110 features. Then, we compute the correlation between these extracted features and the discourse features (e.g., number of lexical chains, unigrams, bigrams, etc.) described previously in order to discard the correlated ones. We consider a p-value with a threshold of 0.8. Then we preprocess the text essays by removing the usernames, Nan values, punctuation and stop words.

**Evaluation Metric**

Following the prior works, we use the Quadratic Weighted Kappa (QWK) to evaluate the performance of our proposed method. It measures the agreement between calculated scores and gold ones.

First, we compute the weight matrix following this formula:

\[ W_{i,j} = \frac{(i - j)^2}{(N - 1)^2} \]

where i,j are the golden scores and calculated scores respectively and N is the number of possible ratings.

Second, we compute the QWK score as follows:
\[ k = \frac{\sum_{i,j} W_{i,j} O_{i,j}}{\sum_{i,j} W_{i,j} E_{i,j}} \]

where \( O_{i,j} \) is the number of essays obtained a rating i by a human annotator and a rating j by the AES system. And the matrix E is calculated as the outer product of histogram vectors of the two ratings. The matrix E is then normalized such that the sum of elements in E.

**Results and analysis**

Table 25 displays the empirical results of our proposed model on the ASAP data as well as the result of other models that have been derived from the literature. We report the QWK scores in each prompt and then we average the scores. As shown in the table, our proposed model outperforms several existing AES systems, in terms of the average score of the Quadratic Weighted Kappa, such as the baselines LSTM, CNN and logistic regression. It also outperforms BERT with an improvement of 3%. The results also demonstrate that incorporating handcrafted features into the deep learning models increase the average performance significantly. Adding features to the LSTM increases the average QWK score from 0.55% to 0.72%. We can observe the same performance improvement when we added the discourse features to the XLNET model with 1% improvement in the average QWK score.

<table>
<thead>
<tr>
<th>Model</th>
<th>Prompt1</th>
<th>Prompt2</th>
<th>Prompt3</th>
<th>Prompt4</th>
<th>Prompt5</th>
<th>Prompt6</th>
<th>Prompt7</th>
<th>Prompt8</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM</td>
<td>0.37</td>
<td>0.40</td>
<td>0.51</td>
<td>0.77</td>
<td>0.76</td>
<td>0.76</td>
<td>0.63</td>
<td>0.17</td>
<td>0.55</td>
</tr>
<tr>
<td>CNN</td>
<td>0.80</td>
<td>0.65</td>
<td>0.63</td>
<td>0.76</td>
<td>0.75</td>
<td>0.76</td>
<td>0.75</td>
<td>0.68</td>
<td>0.72</td>
</tr>
<tr>
<td>XLNET</td>
<td>0.77</td>
<td>0.68</td>
<td>0.69</td>
<td>0.80</td>
<td>0.78</td>
<td>0.79</td>
<td>0.78</td>
<td>0.62</td>
<td>0.74</td>
</tr>
<tr>
<td>BERT</td>
<td>0.82</td>
<td>0.39</td>
<td>0.76</td>
<td>0.88</td>
<td>0.87</td>
<td>0.58</td>
<td>0.81</td>
<td>0.54</td>
<td>0.71</td>
</tr>
<tr>
<td>Logistic regress</td>
<td>0.82</td>
<td>0.64</td>
<td>0.66</td>
<td>0.70</td>
<td>0.78</td>
<td>0.67</td>
<td>0.72</td>
<td>0.60</td>
<td>0.70</td>
</tr>
<tr>
<td>LSTM+features</td>
<td>0.80</td>
<td>0.62</td>
<td>0.60</td>
<td>0.77</td>
<td>0.77</td>
<td>0.77</td>
<td>0.76</td>
<td>0.64</td>
<td>0.72</td>
</tr>
<tr>
<td>XLNET+features</td>
<td><strong>0.85</strong></td>
<td>0.65</td>
<td>0.68</td>
<td>0.81</td>
<td>0.78</td>
<td>0.72</td>
<td>0.79</td>
<td><strong>0.73</strong></td>
<td><strong>0.75</strong></td>
</tr>
</tbody>
</table>
In addition, our proposed model reached the highest scores in prompt1 and prompt8 with QWK scores of 0.85% and 0.73% respectively.

**XLNET Encoded Embeddings Interpretation**

To understand the encoded embeddings generated from XLNET, we generated the encoded embeddings of 768 dimensions using the testing ASAP dataset which consists of 270 text essays. Then, we computed the correlation between every feature of this embedding matrix and the Coh-Metrix features extracted from the same dataset. We consider a threshold of p-value of 0.7. We found that every encoded feature correlates with at least one Coh-Metrix feature or 15 features at most. This explains the discourse nature of the encoded features encoded by XLNET.

To avoid this overlap, we have conducted a final experiment by excluding all the Coh-Metrix features from our proposed model.

Table 26 displays the experimental results across all the eight prompts.

<table>
<thead>
<tr>
<th>Prompt</th>
<th>QWK %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt1</td>
<td>0.80</td>
</tr>
<tr>
<td>Prompt2</td>
<td>0.79</td>
</tr>
<tr>
<td>Prompt3</td>
<td>0.78</td>
</tr>
<tr>
<td>Prompt4</td>
<td>0.83</td>
</tr>
<tr>
<td>Prompt5</td>
<td>0.84</td>
</tr>
<tr>
<td>Prompt6</td>
<td>0.80</td>
</tr>
<tr>
<td>Prompt7</td>
<td>0.81</td>
</tr>
<tr>
<td>Prompt8</td>
<td>0.78</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.8073</strong></td>
</tr>
</tbody>
</table>

The above results show an improvement in the average QWK score after discarding the Coh-Metrix features from the model yielding the state-of-the-art results on the ASAP dataset.
Conclusion

As an extension of the short student answers, we tackle the problem of Automated Essays Scoring. To provide constructive feedback to the learner about the aspect of the essay that needs improvement, we assessed the discourse aspect of the essay which has not been explored extensively in the literature. Motivated by the successes of the hybrid approach in the AES task, we proposed a hybrid method which consists of extracting the encoded features from the essays using XLNET model and concatenating them with handcrafted features that capture the discourse aspect of essays in terms of cohesion and coherence. We have conducted several experiments including and excluding the Coh-Metrix features. The experimental results demonstrated the effectiveness of our approach yielding the state-of-the-art results.

In the future, we are planning to overcome the shortcoming of XLNET in processing the longer sequences more than 512 tokens. In Prompt 8, the average length of all essays is about 650 words, which is larger than the limit. Discarding some important tokens may negatively affected the performance. We believe taking this direction may improve the current QWK results.
Chapter 8

Deep Knowledge Tracing using Temporal Convolutional Networks

Introduction

Modeling students’ knowledge states reflecting their level of mastery with respect to a domain or set of targeted skills and concepts as well as predicting their future performance is an important task in learning science and engineering and is known as the Knowledge Tracing task (KT). It is usually leveraged to optimize students’ learning trajectories, experiences, and outcomes. KT is a challenging task due to the complexity of the human learning processes (e.g., guessing, forgetting etc.) and the inherent difficulties of modeling knowledge (e.g., prior background; (Piech et al., 2015)). Further improvements in KT, which is the focus of our work, will have a wide range of benefits including better adaptation to individual learner’s needs and, consequently, improved effectiveness at inducing learning gains and better learning experiences. There are other benefits of better KT solutions, as exemplified next. Given the tight link between domain models, i.e., the set of concepts to be mastered in a target domain, and better KT models will inform the refinement of domain models. In addition, better KT will lead to designing new, more effective learning materials and instructional strategies.

Given the success of deep neural networks in other domains, deep KT models have gained significant attention recently. They use deep learning techniques to represent learners’ latent knowledge states using large vectors of “artificial neurons”. The parameters of these vectorial representations are inferred from data. Existing Deep KT models use one hot encoding of the identification numbers (IDs) of the instructional activities, e.g., questions or problems to be solved, ignoring those items’ characteristics. Several research works demonstrated that incorporating the items’ semantics in the form of text embeddings into deep KT models can boost their prediction
performance (Sonkar et al., 2020). Despite the state-of-the-art results obtained by these models, there is room for improvement. For instance, many of the existing deep KT models do not account for other important information such as the number of the correct attempts to solve a task and the duration of each step, which can be viewed as indicators of levels of engagement. To address these limitations and enhance the capabilities of KT models, many researchers incorporated into deep KT models various components such as learning ability (Minn et al., 2018), prior knowledge (Shen et al., 2020), and slipping and guessing effects (Cheng et al., 2020). Furthermore, existing deep KT models consider all questions/items under a specific concept as equivalent observations. In attempt to improve instruction adaptation (Shailaja et al., 2014) which will maximize (effectiveness) and speedup (efficiency) students’ learning, we propose a generic framework that explores the underlying information among questions to enhance the performance of KT. The key components of our framework are:

- an NLP embedding component using the Sentence Universal Encoder (Cer, D.M. et al., 2018). Given a question, we extract the semantics associated with its knowledge components (KCs) by averaging the embeddings corresponding to the textual descriptions of these knowledge components.

- a component capturing the engagement level of the student by dynamically assigning students into various groups based on their frequency interactions data using K-means clustering. To the best our knowledge this the first time an engagement level component is included in a deep KT model.

- a question/item difficulty component.

- knowledge embeddings resulted from the fusion of all these inputs (engagement level, text embeddings and item difficulty), using concatenation. The embeddings should
lead to similar representations for items targeting similar concepts and having similar difficulty levels.

- an LSTM-based model to infer the students’ latent knowledge states and a Temporal Convolutional Network to predict future responses.

We conducted series of experiments on the Cognitive Tutor datasets (Stamper et al., 2010) and we compared our proposed model to existing deep KT models.

**Related Work**

Several researchers incorporated important information about questions/items that can help better capture the wider learning context and therefore lead to improved ways to solve the KT problem. For example, Liu and colleagues (Liu et al., 2020) proposed a Pre-training Embeddings via Bipartite Graph (PEBG) to learn a low dimensional embedding for each question based on additional information including question difficulty, question similarity, and skill similarity. They also introduced a product layer to fuse all the input features and obtain the final question embeddings which are incorporated into existing deep KT models. Experiments indicated an AUC (Area Under the Curve) improvement over state-of-the-art results by 8.6% on average. Ghosh and colleagues (Ghosh et al., 2020) proposed an attentive knowledge tracing (AKT) model which combines an attentive neural model with various novel and interpretable model components inspired by cognitive and psychometric models. In addition to the question similarity, AKT uses Rasch model parameters: question difficulty and learning ability. The goal was to learn question embeddings that capture individual differences among questions targeting the same concept. Experimental results demonstrated improved performance of the AKT over prior KT models with a reported AUC improvement of up to 6%. To better model the individualization of prior knowledge and learning rates of various students, Shen, and colleagues (Shen et al., 2020) proposed
a novel Convolutional Knowledge Tracing (CKT) model. More specifically, Hierarchical Convolutional layers were designed to extract learning rate features by processing many continuous learning interactions within a sliding window. Individualized prior knowledge was assessed according to students’ historical learning interactions. Yang and colleagues (Yang et al., 2020) argued that knowledge tracing is affected by the most recent questions answered by students according to the forgetting curve theory. Based on this assumption, they proposed a Convolutional Knowledge Tracing (CKT) model that captures the long-term effect of the entire question-answer sequence and short-term effect of the recent questions using 3D convolutions. The experimental results showed an AUC improvement relative to existing KT models.

Cheng and colleagues (Cheng et al., 2020) proposed an adaptable knowledge tracing (AKT) framework that integrates slipping and guessing factors into the model to obtain more reasonable knowledge state results and leverage the semantics of question texts for more precise knowledge tracing. They obtained improved performance over several KT models.

Shin and colleagues (Shin et al., 2020) proposed SAINT+, a successor of SAINT which is a Transformer based knowledge tracing model. The main addition in this new version is incorporating two temporal features: elapsed time which is the time taken by the student to answer a question and lag time, the time interval between adjacent learning activities. The empirical results showed an improvement in the AUC over the SAINT model on EdNet, the largest public dataset in the education domain according to some metrics, e.g., data from almost 800k students. It should be noted that EdNet contains learning data from the domain of English learning, i.e., it does not target learning of complex STEM topics.

In a continuous effort to overcome the major limitation of deep KT models in capturing differences among questions targeting the same concept and to enhance the knowledge tracing
ability, we consider, as already noted, additional information such as students’ engagement level. We also consider the semantics of the key concepts targeted by each question using a novel natural language processing (NLP) algorithm, i.e., the Universal Sentence Encoder. We also employ for the first time a novel times series model based on a Temporal Convolutional Network to predict student performance.

**Model Architecture**

Our proposed framework (see Figure 21) comprises several important components.

![Model Architecture Diagram](image)

Figure 21. Model Architecture

First, we account for the engagement level of students in the form of a one hot encoded vector of the engagement cluster that the student belongs to at each time t. Second, our model uses an averaged embedding vector of the knowledge components (key concepts) associated with each question. The third component consists of a one hot encoded vector reflecting the difficulty level of each question. Fourth, we use a one hot encoded vector of the ID of each question. Then, we
fuse all these inputs to generate a knowledge embedding for each question answered by the students. The sequence of these knowledge embeddings of each student are passed to an LSTM based model to learn students’ knowledge states for specific concepts. The sequences of these vector are then passed to a Temporal Convolutional Network (TCN) connected with a Sigmoid layer to predict performance.

**Dynamic Assessment of the Engagement Level of Students using Clustering**

Several research studies have revealed the positive correlation between student engagement and academic performance with higher engagement level associated with better grades (Casuso-Holgado et al., 2013; Lee, 2014). Therefore, we take into consideration this information in predicting the future student performance. Inspired by the frequency-based metrics proposed by Reid et al (Reid et al., 2012) to assess engagement level, we propose to use the following metrics available in the Cognitive Tutor dataset:

- **Step Duration**: the elapsed time of the question in seconds, calculated by adding all of the durations for transactions that were attributed to the question.
- **Incorrects**: total number of incorrect attempts by the student on the question.
- **Corrects**: total correct attempts by the student for the question.
- **Hints**: total number of hints requested by the student for the question.

Based on these metrics, we dynamically assess students’ engagement level by clustering, where each cluster represents a level of engagement. Assigning students into a group with similar engagement level at each time step is performed by k-means clustering. Following the research results of Moubayed and colleagues (Moubayed et al., 2020), we consider a three-level model to classify the students’ engagement levels. Hence, the parameter k of the k-means clustering algorithm is set to 3.
After clustering students into three groups with distinct levels of engagement, the clustering results are added to the training data. That is, given a student with a specific concept (KC), we add its corresponding cluster value (e.g., 1, 2 etc.). Then, a student’s engagement level is encoded as a vector $e_t$ with $C+1$ dimensions where $C$ is the number of concepts or knowledge components in the dataset. We encode in this vector, the associated concepts, and the engagement level cluster that the student belongs to at each time step $t$ (an instance in the data records) as following:

$$e_t[kc_t] = 1 \text{ and } e_t[C + 1] = c_m$$

where $kc_t$ is the associated knowledge component or concept for the question $q_t$ at a specific time $t$ and $c_m$ is the engagement level cluster where $c_m \in \{0, 1, 2\}$.

![Figure 22. Engagement level evolution of a student at different time steps while interacting with Cognitive Tutor.](image)

As an example, Figure 22 illustrates the evolution of the engagement level of one student at different time steps on three knowledge components: “simple fractions”, “multiplication”, and “combine like terms”. Each bar in the figure represents a different KC. The red color reflects that the student belongs to the low engagement group, blue color reflects belonging to the medium
engagement group, and gray color reflects their belonging to the high engagement group. It is important to emphasize that the engagement level of the student differs from a concept to another.

**NLP Embedding**

To enhance the knowledge tracing ability and improve the prediction of our proposed deep KT model, we add an NLP embedding component that captures the semantics of the knowledge components. To this end, we use the recently proposed NLP embedding approach called Universal Sentence Encoder (USE) [Cer, D.M. *et al.*, 2018]. USE employs the encoder component of the transformer.

![Figure 23. The architecture of the Universal Sentence Encoder](image)

As shown in Figure 23, first, each textual description of a concept (e.g., Calculate product of two numbers, Identify proper fraction from options etc.) is converted to lowercase and tokenized into tokens using the Penn Treebank (PTB) tokenizer. USE relies on a self-attention mechanism that takes into consideration the token order and its surrounding context for generating each token’s representation. The context-based token representations are then converted to a fixed length text vector by computing the element-wise sum of the representations at each token position. The
encoder outputs a 512-dimensional vector for the knowledge component text embedding. Then, we average the embeddings of the knowledge components of each question at each time step $t$ to capture the semantics of each question with respect to the targeted concepts or KCs.

**Difficulty Constraint**

To capture difficulty of a question, the main assumption we make is that the more the number of mathematical concepts required to answer a specific question the more difficult and sophisticated the question is. Therefore, we calculate the difficulty level of each question by dividing the total number of KCs in a question by the total number of KCs in the Cognitive Tutor dataset.

After calculating questions’ difficulty, we encode the question difficulty as a vector with $C+1$ dimension where $C$ is the number of concepts. Following the prior works, the first $C$ entries of the vector represent the one hot encoded concepts. The last entry of the vector represents the difficulty level of the question.

**Knowledge Tracing using LSTM**

After obtaining the knowledge embedding sequences of each student at time $t$, we pass them to an LSTM base model to compute the knowledge hidden states that represent the mastery level of different KCs.

The resulted hidden states are passed to the Temporal Convolutional Network (TCN) to compute the prediction of future answers.

**Response Prediction using Temporal Convolutional Network (TCN)**

Temporal Convolutional Networks (TCN) have been proposed first by Lea et al (Lea et al., 2016) for the video-based action segmentation task. The distinguishing characteristics of TCNs are:
the convolutions in the architecture are causal, meaning that there is no leakage from future to past.

- the model can take a sequence of any length and map it to an output sequence with the same length.

Added to this, TCN-based models outperform the LSTM-based models for time series predictions.

Motivated by these advantages, we propose to use TCN to improve the prediction of future responses, as described next. Let $H^i$ be the knowledge state with respect to the target domain of student $i$ at time step $t$. The TCN’s architecture is composed of 1D pooling/upsampling and channel wise normalization layers in the encoder. For each of the convolutional layers in the encoder, we apply a set of 1D filters that capture how the input signals evolve over the course of an action. Pooling enables us to efficiently compute activations over a long period of time. The channel wise normalization has been effective in recent CNN methods.

The i-th component of the activation vector output of the encoder is calculated as follows:

$$E^l_{i,t} = f\left(b^{(l)}_i + \sum_{t'=1}^{d} \langle W^{(l)}_{ti}, E^{(l-1)}_{t+d-t'} \rangle \right)$$

where $f(.)$ is a Leaky Rectified Linear Unit, $b^l$ represents the biases, $E^{(l-1)}$ is the activation matrix from the previous layer, and $d$ is the filter duration, and it is set as the mean segment duration for the shortest class from the training set.

The pooled activation vector $\hat{E}^l_t$ is then normalized by its highest response at that time step $m = max_i \hat{E}^l_{i,t}$ with small $\epsilon$ such that:
$$E_t^{(l)} = \frac{1}{n+\varepsilon} \hat{E}_t^{l}$$

The decoder part of the TCN is similar to the encoder and the order of the operations is the following: upsample, convolve, then normalize. The activation output is $D_t^{(l)}$.

Finally, we compute the prediction probabilities of answering future questions by passing the activation matrix output of the decoder to a sigmoid function as follows:

$$Y_t = \text{sigmoid} \left( D_t^{(l)} \right)$$

**Experiments**

We conducted several experiments to demonstrate the effectiveness of our proposed model for the knowledge tracing (KT) task using the Cognitive Tutor datasets. Details about the data are provided next.

**The Data**

Our datasets come from the Cognitive Tutor (Anderson et al., 1995) that teaches students algebra (middle school and high school). Cognitive Tutor presents a problem to a student in form of questions (also called steps) of many skills/concepts. That is every question targets multiple KCs. The Cognitive Tutor uses Knowledge Tracing to determine when a student has mastered a skill.

In this work, we consider the following attributes of each record in the dataset: Student ID, Step Name that represents the question, Step Duration (sec), KC(Default/SubSkills) that represents the associated skill or concept, Incorrects, Corrects, Hints, and Correct First Attempt that is a binary attribute and we consider it as the target to be predicted in our model.
We used three of the development datasets from Stamper and colleagues (2010): the “2005-2006 Algebra” dataset, the “2006-2007 Algebra” dataset, and the “Bridge to Algebra 2006-2007” dataset. The Algebra I dataset consists of 813,661 total responses over 387 skills covering practice attempts for 3,310 students. The Bridge to Algebra dataset contains data from 1,146 students and includes 3,679,199 total logged responses for 494 skills.

Table 27 details important statistics of the data after preprocessing.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Records</th>
<th>#Students</th>
<th>#Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algebra I 2005-2006</td>
<td>809694</td>
<td>574</td>
<td>113</td>
</tr>
<tr>
<td>Algebra I 2006-2007</td>
<td>2270384</td>
<td>1338</td>
<td>492</td>
</tr>
<tr>
<td>Algebra to Bridge</td>
<td>3679199</td>
<td>1146</td>
<td>494</td>
</tr>
</tbody>
</table>

**Experimental Setup**

**Data Preprocessing**

Consistent with data preprocessing in prior works, we conducted the following data preprocessing steps: removed duplicate records, removed records with NaN KCs, removed records with dummies KCs, discarded learners that have fewer than 10 interactions with the system and discards skills answered by less than 10 students.

**Training and Testing**

In all experiments, we perform 5-cross fold validation using a Tesla K80 GPU and a total of 12 GB of RAM. We split all datasets at the student level: at each iteration, 80% of students were used for training and 20% were used for testing. For LSTM, we considered 100 units as the dimensionality of the output space. For the TCN parameters settings, we set the number of filters to use in the convolutional layers to 64. The kernel size was set to 6 and we considered a dilation list = [1,2,4,8,16,32,64]. An Adam optimizer with a learning rate of 0.001 was used and gradients
were clipped to 1.0 to prevent exploding gradients. Due to the large size of our datasets, we considered a small batch size of 5. To evaluate the performance of our model, we customized the Binary Cross Entropy loss since our model gives predictions for all skills/concepts. Given a new student in the testing data, the model predicts her performance in all the concepts in the Cognitive dataset. Thus, to generate the prediction for a specific skill, we take the column-wise dot product between the predictions and a one-hot encoding of the skill.

Our model was evaluated on the test data and the performance of the model is reported using the area under the ROC curve (AUC). AUC measure ranges from 0.5 reflecting a low ability to distinguish from correct and incorrect answers to 1.0 reflecting a perfect discrimination. We compute the AUC by obtaining the prediction of each student in the testing data across all concepts.

**Compared Methods**

To demonstrate the effectiveness of our proposed model, we compare its AUC results with existing methods the reported results on the same Cognitive Tutor data sets. The methods are:

- **DAS3H** (Chofin et al., 2019): incorporates item-skills relationships and forgetting effects.
- **qDKT** (Sonkar et al., 2020): models every learner’s success probability on individual questions over time. qDKT incorporates graph Laplacian regularization to smooth predictions and uses an initialization scheme inspired by the fastText algorithm.
- **DynEmb** (Xu et al., 2020): enables the tracking of student knowledge without the concept/skill tag information that other KT models require.
- **Transformer-based DKT** (Pu, et al., 2020): a Transformer based model that addresses the forgetting issue by accounting for elapsed time. It also uses the questions-skills associations to learn representations of both frequent and rare questions.
Results and Discussion

Table 28 lists the AUC performance results of our proposed model using the three datasets of Cognitive Tutor. Including all the model components: the engagement level, the question difficulty and the knowledge component text embeddings, the model performs better on the Algebra 2006-2007 dataset with an AUC of 96.57%, followed by the Algebra 2006-2007 dataset with an AUC of 92%, and the Bridge dataset with an AUC of 91%. As shown in Table 4, these results reflect a significant performance gain in comparison with existing deep KT methods.

Table 28. The performance results of the proposed model and the ablation experiments results.

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC Result (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL+QD+KCE-Algebra2005-2006</td>
<td>92</td>
</tr>
<tr>
<td>EL+QD+KCE-Algebra2006-2007</td>
<td>96.57</td>
</tr>
<tr>
<td>EL+QD+KCE-Bridge2006-2007</td>
<td>91</td>
</tr>
</tbody>
</table>

Table 29 lists some examples of the results of our proposed framework based on the engagement level and the difficulty of the questions. As shown in the table, the engagement level of the student has more impact on the performance in comparison with the difficulty of the questions. Students with high levels of engagement perform well regardless of the difficulty of the questions. However, it is less probable that the low engaged students perform well especially when the question is difficult.

Table 29. Example of the results in terms of engagement level and difficulty of the question

<table>
<thead>
<tr>
<th>Engagement Level</th>
<th>Difficulty of Questions</th>
<th>Performance</th>
<th>% of testing data</th>
</tr>
</thead>
<tbody>
<tr>
<td>High (1)</td>
<td>Easy (&lt;0.5)</td>
<td>High</td>
<td>13</td>
</tr>
<tr>
<td>High (1)</td>
<td>Easy (&lt;0.5)</td>
<td>Low</td>
<td>6</td>
</tr>
<tr>
<td>High (1)</td>
<td>Difficult (&gt;0.5)</td>
<td>High</td>
<td>13</td>
</tr>
<tr>
<td>High (1)</td>
<td>Difficult (&gt;0.5)</td>
<td>Low</td>
<td>6</td>
</tr>
<tr>
<td>Low (2)</td>
<td>Easy (&lt;0.5)</td>
<td>High</td>
<td>0.4</td>
</tr>
<tr>
<td>Low (2)</td>
<td>Easy (&lt;0.5)</td>
<td>Low</td>
<td>1</td>
</tr>
<tr>
<td>Low (2)</td>
<td>Difficult (&gt;0.5)</td>
<td>High</td>
<td>0.47</td>
</tr>
<tr>
<td>Low (2)</td>
<td>Difficult (&gt;0.5)</td>
<td>Low</td>
<td>2.68</td>
</tr>
</tbody>
</table>
Ablation experiments

To test the impact of each key component of our model on the overall performance, we conducted a series of ablation experiments. The ablation experiments are:

- First, we removed the engagement level component from the model inputs and evaluated the resulting model on the Algebra2006-2007 dataset. The experiments showed that the model AUC has decreased by 1.07%. This is a significant statistical drop in the AUC results reflecting the important of this component.

- Second, we removed the question difficulty from the model inputs. The results showed that the model AUC has decreased by 0.01% which is not statistically significant. Thus, the question difficulty component is not important in the framework.

- Third, we removed the knowledge component text embeddings from the model’s inputs. The AUC of the model has decreased, reflecting the importance of this component. Generally, the Universal Sentence Encoder boosts the prediction ability of deep learning models.

Table 30. Comparison of performance results between our proposed model and the existing DKT methods using the Cognitive Tutor datasets

<table>
<thead>
<tr>
<th>Model</th>
<th>AUC Result (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL+QD+KCE-Algebra2006-2007</td>
<td>96.57</td>
</tr>
<tr>
<td>DAS3H</td>
<td>86</td>
</tr>
<tr>
<td>qDKT</td>
<td>89.5</td>
</tr>
<tr>
<td>DynEmb</td>
<td>86</td>
</tr>
<tr>
<td>Transformer- based DKT</td>
<td>78.4</td>
</tr>
</tbody>
</table>
Naturally, students’ knowledge states evolve over time. To illustrate this evolution process, we generated several visualizations which are more intuitive and user-friendly representations of mastery levels at one moment in time and over time. Figure 24 depicts a student’s knowledge states over a sequence of 91 questions, covering 17 concepts (e.g., identifying units, perform multiplication, simple fractions, etc.), from the Algebra 2005-2006 dataset. At each time step $t$, a knowledge state consists of an activation vector of 8 dimensions. The lighter the color the lower
the mastery level of the student on a specific concept. Figure 25 shows the mastery level of two students on 6 concepts. Students can recognize their poor knowledge points and the ITS makes individual learning schemes by tracing the knowledge state.

Conclusions

In a continuous research effort to enhance the ability of knowledge tracing and improve the prediction of future responses, we proposed a generic framework that includes several important components. First, the model assesses dynamically the engagement level of students across concepts and incorporates this information into the learned knowledge embeddings. Second, it includes the semantics of the knowledge concepts through learning embeddings using a the recently proposed method called the Sentence User Encoder. Third, the model calculates the difficulty of each question and uses this information together with the other inputs to make predictions. Knowledge embeddings of students are then learned by concatenating all these components. Finally, students’ knowledge states’s evolution over time was modeled using an LSTM neural network. These learned sequences of the hidden states is then passed to a Temporal Convolutional Network to predict the future performances of students. The experimental results showed the effectiveness of our proposed model in comparison with existing deep KT methods, yielding high AUC results on Cognitive Tutor datasets. The conducted ablation experiments demonstrated the importance of the engagement level, the text embeddings of the knowledge components and the Temporal Convolutional Network algorithm. Their elimination or substitution led to a decrease in the AUC results. More specifically, the students showing high levels of engagement perform very well for both the difficult (difficulty rate >0.5) and easy questions (difficulty rate <0.5). However, it is less probable that students with low levels of engagement
perform well, especially, when a question is difficult. This probability increases slightly when the question is easy.

In our future work, we will apply this proposed framework on other educational datasets. We will also investigate additional ways to enhance the model’s interpretability and improve its prediction’s performance.
Chapter 9

Conclusion and Future Work

The main contribution of this work is proposing effective and novel algorithms to automatically assess the knowledge of students while interacting with intelligent tutoring systems. In chapter 2, we proposed effective clustering algorithms to assess students’ prior knowledge as a way to balance, on one hand, authoring costs, and adaptiveness and effectiveness, on the other hand. From chapter 3 through chapter 7, we investigated the potential of the deep learning approach in providing an accurate assessment of the natural language students answers. We took the full advantage of deep learning in capturing rich semantic and syntactic features from the text inputs. Thus, enhancing the performance results of the assessment task. In chapter 8, we propose a novel times series model for the knowledge tracing task.

Research Question 1: What are the most effective ways of assessing the knowledge of students within dialogue based intelligent tutoring systems?

Assessing the knowledge state of students plays a vital role in improving the effectiveness of the intelligent tutoring systems because fully adaptive tutoring presupposes accurate assessment (Chi et al. 2001; Woolf 2008). In this work, we process this assessment from three perspectives: assessing the prior knowledge of students, assessing the open-ended natural students’ responses and knowledge tracing.

Assessing the prior knowledge of students within dialog based ITS facilitates the adaptivity of the instruction. Hence, enhancing the tutoring experience for students and optimizing the learning gains. In this context, we propose to cluster students with similar prior knowledge patterns to lower the authoring costs of the system and inform the adaptivity of the ITS.
Assessing the open-ended students’ responses is a fundamental component in an ITS as it can inform the system with the level of understanding of students towards a specific subject and provides the appropriate hints and feedbacks when the answers are incorrect. It is an extremely challenging NLP task due to the variability of student answers and the shortness of some of them. Deep leaning has recently revolutionized the NLP field yielding the state-of-the-art results in multiple tasks and benchmarks. The great semantic expressiveness, interpretability and performance boost of deep learning networks are the main motivation behind applying them on the students answers assessment task.

Tracing the mastery level of students on a specific topic over time is a formative assessment of the knowledge of students. In this work, we propose a generic framework that incorporates the engagement level, the questions semantics, and the difficulty of questions into a deep knowledge tracing model that learns the hidden knowledge states of students over time and predicts their future performances.

**Research Question 2: How can we achieve a tradeoff between the adaptivity and authoring costs within intelligent tutoring systems?**

Clustering students based on their prior knowledge allows to identify students’ groups and analyze them based on their misconceptions and mastered concepts. The identified groups could then be used to inform the authoring of instructional tasks and within-task instructional strategies and feedback for each group as opposed to each learner, which would be a much more expensive process. To this end, we have applied several clustering algorithms such as DP-means, K-means, Agglomerative clustering, and K-modes algorithm using a data that consists of pretest numeric and text answers collected from 264 high-school students. The experimental results showed the effectiveness of these applied algorithms. Three distinct groups of students have converged: i)-
high knowledge level group, ii)-medium knowledge level and iii)- low knowledge level. The prior knowledge level of each cluster was consistent with the post knowledge level.

We plan in the future to further investigate the resulting clusters for a better understanding of the characteristics of the students in each cluster. Such information can provide information about the major misconceptions of students in each class struggle with. This information can be shared with teachers to help them better plan their lessons plans to address major misconceptions their students may have.

**Research Question3:** *Can a Capsule Network-based model improve the assessment of open-ended natural language answers?*

In this dissertation, we proposed a Bi-GRU-Capsnet model to assess the correctness of the students answers within the intelligent tutoring system DeepTutor. We have chosen this deep learning model to get benefits from its no requirements of hands -crafted features and external resources. Added to this, Capsule networks have the capability to express the semantic meanings in a wider space using a vector that captures the instantiation parameters of the input such as the order of words and their semantic representation. Our proposed model is composed of several important components: an embedding layer, a Bi-GRU layer, a capsule layer and a SoftMax layer. We have conducted several experiments considering a binary classification task: correct or incorrect answers. The experimental results show that our model reached the state-of-the-art results on the DT-Grade dataset at the time of publication. Particularly, our model reached the highest accuracy when using the ELMo embeddings. The major limitation of this proposed model is its inability of assessing the very short answers and resolving the reference matching problem.
In the future, we plan to investigate more deep learning models to enhance the assessment results and overcome the main limitations of the Bi-GRU-Capsnet.

**Research Question 4:** *Can an Attention-based Transformer model improve the performance results of assessing the short students’ answers?*

In this dissertation, we proposed an Attention-based Transformer to assess the correctness of student answers freely generated by students in dialog based ITS. The model has been chosen due to promising results that transformers have achieved in various NLP tasks. Furthermore, attention allows deep learning models to semantically process short texts effectively using the most relevant words in the text. Our proposed model is composed of several important components: an embedding layer, a positional encoding module, a transformer layer and a SoftMax layer. Experimental results on the DT-Grade dataset show high competitiveness of the proposed model rivalling previously proposed state of the art methods. Added to this, our proposed model leveraged an accurate assessment for the very short answers.

In the future, we plan to further explore additional novel deep learning models to improve the current assessment results on the DT-Grade dataset.

**Research Question 5:** *Can a Graph Convolutional Network be utilized to achieve an accurate assessment of open-ended students’ answers?*

Graph Convolutional Networks have achieved impressive results in multiple NLP tasks such as text classification. However, this approach has not been explored yet for the student answer assessment task. In this dissertation, we propose to use Graph Convolutional Networks to automatically assess freely generated student answers within the context of dialogue-based intelligent tutoring systems. We convert this task to a node classification task. First, we build a
knowledge graph where each node represents a concept in Physics whereas the edges represent the relatedness between nodes. Second, the graph is fed to two layers of Graph Convolutional Networks that computes the nodes embeddings. Finally, the output of the second layer is fed to a SoftMax layer for classification. The empirical results showed that our model reached the state-of-the-art results by obtaining an accuracy of 73%.

In the future we plan to explore novel deep learning models that perform well in low resource scenarios such as ours.

**Research Question 6:** Can finetuning the pretrained transformers improve the current performance results of the short students answers assessment downstream task?

Transfer learning has been effective for low resource scenarios. The Pretraining-Finetuning paradigm has revolutionized the NLP field yielding state-of-the-art results. Motivated by these successes, we fine-tuned the pretrained BERT, RoBERTa, DistilBERT, ALBERT, XLNET and T5 on the short students answers assessment downstream task. The experimental results have demonstrated the great effectiveness of this approach yielding state-of-the-art results with a significant improvement of 8%-15% in accuracy over the previous methods. Particularly, T5 model has achieved an accuracy of 88%.

In the future, we are planning to find better pretraining and finetuning strategies such as pretraining these models on domain related data, layers wise Learning Rate and others.

**Research Question 7:** How can we effectively assess the discourse aspect of long essays?

Automated Essays Scoring (AES) is an important educational application in NLP. Most of the research work done in this area is based on a holistic approach which summarizes the quality of an essay with a single score. The drawback of this approach is its inability to inform the learner of
what aspect of the essay needs improvement. To alleviate this limitation and get benefit of the success of the hybrid approach in the AES task, we have proposed a discourse based XLNET model. First, we generate a distributed representation of an essay using the pretrained XLNET. Then, we concatenate this representation with handcrafted discourse features derived mainly from lexical chains and the Coh-Metrix tool. The experimental results demonstrated the effectiveness of this proposed model on the ASAP dataset yielding state-of-the-art results with an average Quadratic Weighted Kappa score of 80.3%.

In the future, we are planning to address the major limitation of XLNET in processing the longer sequences with more than 512 tokens. For instance, the average length of prompt 8 in the ASAP dataset is 650. We believe that including all the essay’s tokens in the final distributed representation of the essay can improve the prompt ‘s QWK score.

**Research Question 8: How can we enhance the knowledge tracing capability and performance?**

Despite the good performance of deep knowledge tracing models, they suffer from several limitations. For example, they ignore pertinent information such the prior knowledge of students, the forgetting and slipping effects, difficulty of questions etc. These factors can improve the performance of the knowledge tracing. To overcome this limitation, we proposed a generic framework that accounts for the engagement level of students, embeddings of the knowledge components that capture rich semantics, and the difficulty of questions. These inputs are used to construct the initial knowledge embeddings of students by concatenating them. Then, we passed these embeddings sequences to an LSTM that learns the hidden states of the knowledge of students. Finally, we passed these hidden states of knowledge to a Temporal Convolutional Neural Network to predict future performances. Several experiments have been conducted using the Cognitive dataset to evaluate the proposed framework. The
empirical results demonstrated the superior performance of this proposed framework over several existing methods in the literature. An AUC of 96.57% has been achieved on the Algebra 2006-2007 dataset.

In the future, we will apply this proposed framework on other educational datasets. We will also investigate additional ways to enhance the model’s interpretability and improve its prediction’s performance.
References


Ait Khayi, N., & Rus, V. & Tamang, L (2021). Deep Knowledge Tracing using Temporal Convolutional Networks. AI4EDU@IJCAI.


Beck, D., Haffari, G., & Cohn, T. (2018). Graph-to-Sequence Learning using Gated Graph Neural Networks. ACL.


Chen Zhang, Qiuchi Li and Dawei Song. (2019). Aspect-based Sentiment Classification with Aspect-specific Graph Convolutional Networks. In the proceedings of the Empirical Methods in Natural Language Processing (EMNLP)


Nicola De Cao, Wilker Aziz and Ivan Titov (2019). Question Answering by Reasoning Across Documents with Graph Convolution Networks. In proceeding of NAACL-HLT


Persing, I.; and Ng, V. (2014). Modeling Prompt Adherence in Student Essays. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1:


Pushkar Mishra, Marco Del Tredici, Helen Yannakoudakis and Ekaterina Shutova (2019). Abusive Language Detection with Graph Convolutional Networks. In proceedings of NAACL-HLT.


Yu Cao, Meng Fang and Dacheng Tao (2019). BAG: Bi-directional Attention Entity Graph Convolutional Network for Multi-hop Reasoning Question Answering. In the proceedings


