Modelling Alignment and Key Information for Automatic Grading

Author: Suraj Jung PANDEY

Supervisor: Dr. Alistair WILLIS
Dr. Paul PIWEK

A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

Natural Language Processing Group
School of Computing and Communications

Jan 6, 2022
Declaration of Authorship

I, Suraj Jung Pandey, declare that this thesis titled, “Modelling Alignment and Key Information for Automatic Grading” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.

- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.

- Where I have consulted the published work of others, this is always clearly attributed.

- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

- I have acknowledged all main sources of help.

- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
Signed:
Suraj Jung Pandey

Date:
31/12/2021
Automatic Short Answer Grading (ASAG) is the task of grading short answer questions using computer models, especially machine learning and artificial intelligence. Short answer questions differ from essay questions, as the required short answer response is usually about a sentence in length, whereas for essay questions, students are expected to elaborate in detail on the subject. Also, unlike multiple-choice questions, for short answer questions students are not provided with choices from which to select a correct response. For short answer questions, students write the response in free-text or natural language. Since students can formulate the response in variety of ways, a single short answer question can have multiple forms of correct response. This makes the ASAG task a very challenging one.

This dissertation introduces the notion of key information for semantic text similarity in the context of ASAG. We define key information as new information that is not present in the question, and which is essential for the answer to be correct. We propose an algorithm for identifying key information and develop a neural structured alignment model which uses the key information to accurately grade student responses. We test the model on the Beetle and SciEntsBank corpora against a range of state-of-the-art models, with our model outperforming the state-of-the-art models.

The primary contribution of this dissertation is a set of methods that use semantic similarity techniques to obtain state-of-the-art results in the ASAG task. We first investigate how well existing neural network models work on the ASAG task. We look at word alignment and structured alignment attention networks. We then show how the performance of these architectures can be improved by augmenting them with key information. I present two different approaches to learning key information. The first approach uses an intuitive algorithm to learn the key information from a question’s reference answers. The key information is then passed to the neural network as
input. The second approach uses a complex-number model to learn key information within the neural network itself.
Acknowledgements

Firstly I would like to thank my supervisors Dr. Alistair Willis and Dr. Paul Piwek for their constant support. They have been great sources of inspiration and encouragement. Their advice and assistance always got me back on track whenever I was confused and astray.

A special thanks go to Dr. Patricia Charlton who kept me motivated with off-PhD conversations and work.

I owe a great deal to my friends in England. Matt, Sonia, Janka, Yadu, Telma, Paco, Jacopo, Suresh, and Binita: you guys helped a lot to make this journey a pleasant one.

In every ups and down my constant source of happiness has been my partner Nitu Bharati. I cannot thank you enough.

I would also like to acknowledge the generous support of the Open University for providing the funding and infrastructure including invaluable PG-forum.

Finally, unconditionally supporting me always although far is my family back in Nepal Dad, Mum, Nisha, Jyoti, Gaurav Bhinaju, and Katha. I love you all.
## Contents

Declaration of Authorship iii  
Abstract v  
List of Figures xiii  
List of Tables xvii  
Glossaries xix  

### 1 Introduction 1  
1.1 Scope of Automatic Short Answer Grading 2  
1.2 Automatic Short Answer Grading (ASAG) 4  
1.3 Research Question 8  
1.4 Contribution 10  

### 2 Background 13  
2.1 Introduction 13  
2.2 Methods for ASAG 15  
2.2.1 Use of Templates for ASAG 16  
2.2.2 Alignment for ASAG 20  
2.2.3 Machine Learning for ASAG 23  
2.2.3.1 Background on Semantic Similarity 23  
2.2.3.2 Semantic Textual Similarity systems 26  
2.2.3.3 Use of Similarity in Automatic Grading 30
3.7 Conclusion ................................................................. 103

4 Incorporating Key Information for ASAG .......................... 105
4.1 Introduction .............................................................. 105
4.2 The importance of Key Information for Automatic Grading . 108
4.3 Key Information Extraction (Extractor) ......................... 110
4.4 Key information model (SACA + KI) ......................... 113
4.5 Evaluation .............................................................. 117
  4.5.1 Evaluation Setup ................................................. 117
  4.5.2 Results .............................................................. 118
4.6 Conclusion .............................................................. 122

5 ASAG Based on Complex-valued Word Embeddings ........... 125
5.1 Introduction .............................................................. 125
5.2 Related work ........................................................... 126
5.3 Methodology ............................................................ 128
  5.3.1 Complex Word Embeddings .................................. 128
  5.3.2 Density Matrix Representation of Words .................. 129
  5.3.3 Interaction Between Density Matrices ....................... 130
  5.3.4 Neural Network for ASAG Using Complex Embeddings 131
  5.3.5 A Baseline Comparison Based on Multi-Tasking Learning (MTL-ASAG) ........................................ 133
5.4 Experiments ............................................................. 134
  5.4.1 Method: Evaluation Setup ..................................... 134
  5.4.2 Results .............................................................. 135
5.5 Conclusion .............................................................. 138

6 Conclusion ............................................................... 139
6.1 Research Questions and Findings Revisited ..................... 139
6.2 Limitation ............................................................... 141
List of Figures

1.1 Sample Multiple Choice Question (MCQ). Students select an answer from the given choices. ........................................ 5

2.1 A simple mark scheme template (Mitchell et al., 2002) ............. 17
2.2 Alignment between chunks in a reference answer (top) and a student response (bottom). ........................................... 20
2.3 Steps for alignment as proposed by Sultan, Bethard, and Sumner (2014) 28
2.4 Example of alignment with dependency parse. .......................... 29
2.5 Feed forward neural network. ................................................ 48
2.6 (Kenter and De Rijke, 2015) Hypothetical example — two-dimensional representation of the word embeddings for two short texts (each consisting of three terms), represented as transparent and opaque dots respectively. The corresponding means of the two sets of embeddings are depicted as crossed circle .................. 53
2.7 A typical setting for matching two sentences in a deep learning framework ................................................................. 56
2.8 A convolution neural network with four filters and max-pooling. ... 57
2.9 A RNN network on the left and its unrolled form on the right ....... 59
2.10 An example alignment between reference answer (top) and correct student’s answer (bottom) ........................ 61

3.1 An example alignment between the reference answer (top) and the correct student’s response (bottom) .................. 66
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>An example alignment between the reference answer (top) and the correct student’s response (bottom).</td>
</tr>
<tr>
<td>3.3</td>
<td>An example alignment between a reference answer (top) and an incorrect student answer (bottom) with word level alignment.</td>
</tr>
<tr>
<td>3.4</td>
<td>An example alignment between a reference answer (top) and an incorrect student answer (bottom) with word level alignment. The dashed arrow shows a contradictory pair.</td>
</tr>
<tr>
<td>3.5</td>
<td>A structural representation within a sentence. The structural representation is an example and not a standard parse. The structure depicts the relation of <em>gap</em> with other words in the sentence.</td>
</tr>
<tr>
<td>3.6</td>
<td>Structural alignment between a reference answer (top) and a student response (bottom). The alignment is very weak, thus signifying a weaker relation between two sentences.</td>
</tr>
<tr>
<td>3.7</td>
<td>Structural alignment between a reference answer (top) and a student response (bottom). The alignment is very weak due to contradicting elements in the structure, signifying a weaker relation between two sentences.</td>
</tr>
<tr>
<td>3.8</td>
<td>Embedding a reference answer on the left and student response on the right.</td>
</tr>
<tr>
<td>3.9</td>
<td>Encoding both the reference answer and the student response using a feed forward neural network to tune the vector to the current dataset.</td>
</tr>
<tr>
<td>3.10</td>
<td>Obtaining an unnormalised attention weights matrix by the dot product of encoded vectors.</td>
</tr>
<tr>
<td>3.11</td>
<td>Comparing the aligned pairs.</td>
</tr>
<tr>
<td>3.12</td>
<td>Aggregating the compared pairs.</td>
</tr>
<tr>
<td>3.13</td>
<td>The architecture of attend, compare, and aggregate model from bottom to top.</td>
</tr>
</tbody>
</table>
3.14 Intra-sentence attention. The alignment occurs between words of a single sentence. Here, $u_4$ is the intra-sentence attention representation of $r_4$.

3.15 Random variable representing distribution to compute simple attention. 'S' is the source word for attention.

3.16 Random variable representing distribution to compute a structured attention network. 'S' is the source word for attention.

3.17 Phrase structure parse of a sentence (obtained from AllenNLP (Joshi, Peters, and Hopkins, 2018)).

3.18 Dependency parse of a sentence (obtained from AllenNLP (Dozat and Manning, 2016)).

3.19 Tree structure showing relation between words.

3.20 Generation structure alignment for a sentence using a BiLSTM and the inside-outside algorithm.

3.21 Confusion matrix for ACA and SACA model. The labels 0, 1, 2, 3 and 4 represent correct, partially-correct, contradictory, irrelevant, and non-domain category respectively.

3.22 Normalised attention weights of the decomposable attention model, showing the alignment between the reference answer and the student response. Both heat-maps are from a correct prediction by the model. The left heat-map shows a correct student response while the right heat-map shows an incorrect student response.

3.23 Alignment that highlights the importance of comparison module which aligns $\text{closed}_1 \ \text{path}_1$ with $\text{connected}_2$.

3.24 Alignment map where negation is not detected.

3.25 Alignment that ignores important detail ($\text{change}_1$) from the reference answer.
4.1 A comparison of a student response (bottom) with a reference answer (top), with key information highlighted in bold. The key information assists with more accurate grading. ........................................ 107

4.2 Steps involved in extracting key information from a question’s reference answers. .................................................. 111

4.3 Block diagram for the key information model. The key information is concatenated to the alignment information. .................. 114

4.4 Neural network representation of the key information model. The key information is added to the aggregated alignment information. .................................................. 116

4.5 Weighted F1 score for 2-way classification in development set with varying number of KI. ........................................ 116

4.6 Weighted F1 score for 5-way classification in development set with varying number of KI. ........................................ 117

4.7 Confusion matrix for KI (human) model. The labels 0, 1, 2, 3 and 4 represent correct, partially-correct, contradictory, irrelevant, and non-domain category respectively. .................. 122

5.1 Neural Network model using complex embedding. Complex embeddings are used to create density matrices for both the reference answer and the student response. The densities matrix are matched to obtain value for soft-max. ........................................ 131

5.2 Multi-Tasking Learning model that learns response classification and KI tagging. .................................................. 132
List of Tables

2.1 Similarity scores with explanation (Agirre et al., 2016) . . . . . . . 25
2.2 Number of training cases, unseen answers, unseen questions, and
unseen domains in the corpus. . . . . . . . . . . . . . . . . . . . 34
2.3 Label distribution. Percentages in parentheses. UA, UQ, UD corre-
spond to individual test sets. . . . . . . . . . . . . . . . . . . . . 34
2.4 Weighted average $F_1$ scores for different systems in the SemEval
challenge (Dzikovska et al., 2013a; Sahu and Bhowmick, 2019) . . 41
3.1 Weighted F1 score obtained with the best model in development set
and comparison with development set weighted F1 obtained from
CoMet and Riordan et al. (2017). . . . . . . . . . . . . . . . . . . . . . 94
3.2 Weighted F1 for the 2-way classification for the ASAG models. . . 95
3.3 Weighted F1 for the 5-way classification for the ASAG models . . 95
3.4 Precision and Recall values for ACA and SACA models for 2-way
and 5-way classification. . . . . . . . . . . . . . . . . . . . . . . . . . 96
4.1 Weighted for the 2-way classification for KI and other models . . . 118
4.2 Weighted F1 for the 5-way classification for KI and other models. . 119
4.3 Precision and Recall values for KI(machine) and KI (human) models
for 2-way and 5-way classification. . . . . . . . . . . . . . . . . . . . 119
5.1 Overall Accuracy for the 2-way classification for complex embed-
ding and other models . . . . . . . . . . . . . . . . . . . . . . . . . . . 135
5.2 Overall Accuracy for the 5-way classification complex embedding and other models. ........................................... 135
5.3 Precision and Recall values for Complex Embedding models for 2-way and 5-way classification. .............................. 136
Glossary

align candidate  A word/phrase from a sentence that pairs with another align candidate from a different sentence. 67

aligned pair  Two align candidates linked due to semantic relatedness. 69

alignment  The task of pairing semantically related words/phrases. 66

automatic assessment  Using computer models to grade student responses. 23

reference answer  An answer to a question provided by the question setter or an expert in the field. 7

representation learning  A set of methods that can automatically learn representations from data. 47

SemEval dataset  A dataset formed from the combination of Beetle dataset and Sci-EntsBank dataset. 33

student response  A response to a question provided by a student. 7
Acronyms

ASAG  Automatic Short Answer Grading. v

CNN  Convolution Neural Network. 56

GAN  Generative adversarial network. 128

HMM  Hidden Markov Model. 125

KI  Key Information. 106

LDA  Latent Dirichlet Allocation. 125

LSA  Latent Semantic Analysis. 21, 26

LSTM  Long Short-Term Memory. 91

MCQ  Multiple Choice Question. 3

NLP  Natural Language Processing. 1

PMI  Pointwise Mutual Information. 31

RNN  Recurrent Neural Network. 59

SAQ  Short Answer Question. 3
xxii

**STS** Semantic Textual Similarity. 24
Chapter 1

Introduction

A part of understanding natural language is being able to recognise the similarity in meaning between two or more texts. Identification of semantic similarity between a pair of sentences is an important research problem in Natural Language Processing (NLP) with applications in various areas, such as fake news classification (Ahmed, Traore, and Saad, 2018), machine translation (Mikolov, Le, and Sutskever, 2013), text summarisation (Aliguliyev, 2009), detecting bias in news (Hamborg, Donnay, and Gipp, 2019), textual entailment (Zanzotto, Pennacchiotti, and Moschitti, 2009), question answering (Jeon, Croft, and Lee, 2005), and short answer grading (Mohler and Mihalcea, 2009).

This dissertation focuses on one such research problem: short answer grading. Automatic grading has become an important aspect of the current online education world, where a single course can accommodate hundreds of thousands of students. For instance, Coursera\footnote{https://www.coursera.org/learn/machine-learning#faq: accessed 20/05/2020} has more than 3 million students registered at a time. As the number of students on online modules increases, the cost of manual marking can become prohibitive. In this dissertation, I focus on using semantic similarity approaches to automatically grade student responses by comparing the student responses with one or more model answers provided by domain experts. The methods of grading student
responses presented in this dissertation are modelled and evaluated on a dataset of student short answers. The dataset includes assessments on the introduction to electronics and science module for 3rd to 6th-grade students. We focus on methods that use semantic similarity concepts to grade short responses. Such methods can be used in a variety of settings where reference answers are available to compare with the student responses. Automatically assigning grades to short natural language answers is termed *Automatic Short Answer Grading* (ASAG).

This chapter introduces the task of automatic short answer grading and explains its scope. The chapter also discusses the research questions and their associated studies, along with the contributions. As an added note, the glossary on Page XVII in the dissertation defines the terms that I introduce in the thesis. The numbers in the red font at the end of each glossary and acronym item description is the page number of the first occurrence of the item.

### 1.1 Scope of Automatic Short Answer Grading

Enabling global access to high-quality education at scale is one of the major challenges in education (Woolf et al., 2013). The recent advances in computer-assisted approaches could enable open access to world-class instruction and a reduction in the growing cost of learning. We have seen a practical demonstration of computer-assisted approaches during the lockdown imposed due to COVID-19\(^2\). Although the lockdown meant that students could not attend classes in person, their education could continu through online portals such as Zoom\(^3\). A virtual classroom can be created in Zoom and both learners and educators can access the virtual classroom through a computing device with a microphone, camera, and access to the internet.

\(^3\)https://zoom.us/
The experience in the lockdown showed that we can in practice make high-quality education accessible to a large body of learners. A virtual classroom showed that a person with a computer and an internet connection can access the classroom from any place in the world. Additionally, the number of learners is not limited to the size of a physical classroom but can expand to a huge size (millions for Coursera as mentioned in Section 1) depending upon the computing infrastructure.

However, a major barrier to this task has been the need to automatically provide meaningful and timely feedback on student work. Although with computer-assisted approaches a single educator can provide high quality education to a large group of learners, the cost of providing feedback manually could be prohibitive. For a large online class, it can take a long time for human markers to provide feedback to each of the students. Also, a human marker may not be available round the clock when a student needs quick feedback.

Automatic grading of student responses can be of high value due to the immediacy that automatic grading systems provide, where test-takers would otherwise need to wait for the human marker to complete the grading. Additionally, providing practice exams with instant feedback can help students improve their performance on subsequent exams (Siddiqi and Harrison, 2008). Siddiqi and Harrison showed that there was a significance improvement in the performance of the students in subsequent exams when they were provided with feedback for their previous exam, compared to not having any feedback provided.

Other forms of assessment like Multiple-Choice Questions (MCQs) have their own benefits. MCQs are a common type of assessment due to their reliability, validity, and ease of scoring (Pepple, Young, and Carroll, 2010). MCQ tests are commonly constructed to assess students’ ability to recall isolated pieces of information rapidly (Oyebola et al., 2000). In contrast, for a Short Answer Question (SAQ), students need
to show more flexibility in their response, so that interpretative skills can be evaluated (Feletti, 1980). The freedom to answer questions in their own way can allow students to demonstrate their competence better than with multiple-choice questions (Newble, Baxter, and Elmslie, 1979). We will discuss the benefits of natural language responses in more detail in Section 1.2.

SAQs can however pose some challenges. A short response can have linguistic variations (a given response can be formulated in many different ways) and SAQs have a subjective nature, in that a question may have multiple possible correct answers or no correct answer. These challenges of SAQs may lead to inconsistency in the awarded grade caused by different interpretation by the markers (Roy et al., 2016). ASAG can address this problem because the awarded grade by the automated system will be consistent; a computer model will award the same grade to the same response.

### 1.2 Automatic Short Answer Grading (ASAG)

Assessment is very important for education as it is a judgement of a student’s work. The judgement follows a certain standard and results in comparative or numerical ratings. A student’s work can be made up of a wide variety of tasks, ranging from lab-works, programming exercises, mathematical proofs, essays, MCQs, or questions requiring short natural language response. An essay requires students to elaborate in detail about the subject matter, multiple-choice questions require students to select one or more answers from a provided set, whereas SAQs require students to provide the response in a short passage of free-text, usually just one sentence. The judgement on students’ work can be provided both manually or automatically. In this dissertation, I focus on questions that require a short natural language answer and I will develop a method that can automatically grade such short responses using Natural Language Processing techniques.
1.2. Automatic Short Answer Grading (ASAG)

**Question**

What is the unit of density ($\rho$)?

Tick one box. [1 mark]

<table>
<thead>
<tr>
<th>Joules, J</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Joules per kilogram, J/kg</td>
<td></td>
</tr>
<tr>
<td>Kilograms, kg</td>
<td></td>
</tr>
<tr>
<td>Kilograms per metre cubed, kg/m$^3$</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 1.1:** Sample Multiple Choice Question (MCQ). Students select an answer from the given choices.

The difference between multiple choice and short answer questions is easy to understand. As shown in Figure 1.1 we can see that in MCQ, students are given choices from which they must select an answer. In contrast, for SAQs, such choices are not provided. Students have to recall and formulate the response in their own words. The difference between other question types such as SAQs and essays can become blurred. To avoid ambiguity, we will use the definition of SAQ as proposed by Burrows, Gurevych, and Stein (2015):

**Definition 1.1** Short Answer Question: A short answer question is one that meets at least four specific criteria:

1. The question must require a response that recalls external knowledge, instead of requiring the answer to be recognised from within the question.

2. The question must require a response given in natural language.

3. The answer length should be roughly between one word/phrase and one paragraph.
4. The assessment of the responses should focus on the content instead of writing style.

For example, consider Example 1.1 from the Beetle dataset (Dzikovska et al., 2013a), containing a question and student response.

**Example 1.1**

*Question:* Explain why you got a voltage reading of 1.5 for terminal 1 and the positive terminal?

*Answer:* because there is a gap

In Example 1.1, the answer has new information that is not present in the question (the word *gap*). The answer is written in natural language, it has one phrase and although not a complete sentence, it is a valid elliptical utterance. Additionally, the answer provides the required information and the assessment will allocate marks on the basis of that. Thus the answer can be termed as short answer.

Providing a response in a natural language has cognitive benefits for students (Newble, Baxter, and Elmslie, 1979; McDaniel et al., 2007). The major cognitive benefits of testing in general are that testing enhances learning and retention. The study published in the European Journal of Cognitive Psychology shows that recalling responses in natural language for SAQs has better effect on learning and retention compared to recognising correct answers in MCQs (McDaniel et al., 2007). From the study conducted by the authors, the mean proportions of questions answered correctly indicated that participants were more likely to answer MCQs correctly than SAQs. The authors argue that this suggests that MCQ is a less demanding retrieval task than SAQ. Additionally, the study also showed that on successive assessments, students’ performance were better for the questions that were repeated on the previous assessment on SAQs compared to repeated ones on MCQs. This result shows that
SAQs promotes retrieval processing that is more mnemonically potent than MCQs. However, providing responses in a natural language does pose a significant challenge for ASAG systems. The linguistic variation across the possible correct answers for a question makes such questions harder to grade both for humans and automatic language processing models (Burrows, Gurevych, and Stein, 2015).

In a typical setting, ASAG systems grade student responses by comparing them with one or more reference answers provided by the domain experts or question setters (Jordan, 2012; Mitchell et al., 2002). Example 1.2 is taken from the Beetle dataset (Dzikovska et al., 2013a).

**Example 1.2**

*question:* Explain why you got a voltage reading of 1.5 for terminal 1 and the positive terminal?

*reference answer 1:* Terminal 1 and the positive terminal are separated by the gap

*reference answer 2:* Terminal 1 and the positive battery terminal are in different electrical states

*Student #1 (C):* Terminal 1 and the positive terminal are not connected

*Student #2 (P):* because there is a gap

*Student #3 (I):* The negative battery terminal is separated by a gap from terminal 1

The students’ answers in Example 1.2 are marked as correct (C), partially correct (P), and incorrect (I). Student #1’s response is correct as it contains all the information
present in the reference answer 1 such as terminals and gap. However, the information is provided in the response not connected, which is in a different form from gap, the term in the reference answer. For Student #2, the response is missing the terminal information but still has information about gap and so is partially correct. The challenge here is to recognise that the student response contains enough information for it to be partially correct even though some of the information is missing. Finally, even though Student #3’s answer shows a high amount of similarity with reference answer 1, it contains a contradictory term negative which makes the response incorrect. Thus the ASAG system should also recognise important contradictions that might exist in the response. All of these makes ASAG a challenging and interesting problem.

1.3 Research Question

Having established that ASAG is a problem worth studying we will now look at the research questions that we will answer in this dissertation. We propose the following research questions:

RQ1: How can we use the concept of semantic similarity in the ASAG task?

RQ2: Is the linguistic information encoded within syntactic structure an important signal for ASAG?

RQ3: How can we identify the most important information in an answer to a question, and use it to help decide whether a student response is correct?

RQ1: A core hypothesis underlying the ASAG models proposed in this dissertation is that the grading of a student response can be obtained by semantically comparing a reference answer and a student response. For this, I will follow a four stage approach where the model:
1. divides the student response and reference answer into chunks,

2. aligns the chunks between sentences,

3. estimates the similarity between aligned chunks, and

4. combines the chunk similarity to obtain an overall sentence similarity.

I use a neural network architecture that can take two short sentences as input and then aligns, compares, and aggregates the sentences for the final prediction. This architecture obtains the state-of-the-art result in a standard dataset for ASAG.

Chapter 3 discusses the details of the methods.

**RQ2**: Addressing the first research question results in an alignment method for the ASAG task. Now, how can we improve the alignment task further? To answer this I adopt a method to extract structural relationships within the reference answer and student response and use these relations for alignment. The use of such structural units provides a better semantic comparison than using a word-level model (Montemagni and Vanderwende, 1993).

Linguistically motivated representations of text structure rely on the availability of annotated corpora as well as a wider range of standard NLP tools such as tokenisers, pos-taggers, and syntactic parsers. Unfortunately, the reliance on labelled data, which is both difficult and highly expensive to produce, presents a major obstacle to the widespread use of the structure for text modelling. Moreover, despite recent advances in structure processing, the use of an external parser often leads to pipeline-style architectures where errors propagate to later processing stages, affecting model performance. To this end, there are various techniques to infer structural representations directly from the text without having to access a parser or structure annotated data. I adopt a neural network model that can learn structure within sentences without any training annotation of such structured. Such structures are then used further
in the neural network layer to make a semantic comparison of a reference answer and a student response.

Chapter 4 discusses the details of the methods.

RQ3: The most important information in an answer is the new information that is not present in the question but is necessary for the answer to be correct. This research question has two parts. First, it is essential to automatically identify the important information from the answer without any human annotation. Second, such important information should be seamlessly used in an ASAG system.

To identify the important information, we focus on the setting where more than one reference answer is provided for a question. It is a common practice to provide more than one reference answer in a normal assessment setting. We then work on the hypothesis that important information is present in more than one reference answer in an identical form or a semantically similar form. I formulate an algorithm to extract such repeated terms.

Chapter 5 discusses the details of the important information extraction algorithm.

Once the important information has been identified, we need to use it in an ASAG system. I develop a neural network architecture that integrates important information along with semantic comparison.

Chapter 6 and 7 discuss the details of how we can use the key information in a neural network setting.

1.4 Contribution

The primary contribution of this dissertation is a set of methods that use semantic similarity techniques to obtain state-of-the-art results in the ASAG task. The fine-grained contributions are:
1. A system that uses word-level alignment to automatically grade students’ responses. I use a neural network architecture for ASAG that improves on the previous state-of-the-art ASAG technique.

2. A system that learns and uses structural relations to automatically grade students’ responses. I use a neural network architecture that improves on the word-level alignment for ASAG.

3. An algorithm that identifies the important information from reference answers.

4. A neural network architecture that uses important information in conjunction with the semantic matching between two sentences for ASAG task.

5. Improvements on the state-of-the-art on a standard ASAG dataset.
Chapter 2

Background

2.1 Introduction

Assessment is a variety of techniques used to gather information about students’ performance. Assessments are useful to measure the progress of the student throughout the learning period and to measure the student’s grasp of the knowledge at the end of a course (Taras, 2005; Garrison and Ehringhaus, 2007). The outcome of an assessment is a score awarded to students which is a function of their performance in the assessment. Computer-based assessments are gaining in popularity compared to the pen and paper based assessments. Two key advantages of computer-based assessment are (Reynolds et al., 2010):

1. **Instant output**: Students can receive feedback immediately rather than having to wait for it, possibly for up to several weeks. This also means that assessments can be conducted for large classrooms. In a scenario where a student can take a course remotely though a computer, virtual classrooms sometimes accommodate hundreds of thousands of students.

2. **Adaptive testing**: In adaptive testing, the next question for the student is selected depending on the student’s performance in the current question. For example, if a student’s response to a question is correct, then the next question
will be comparatively more difficult. Conversely, if the response is incorrect then the next question will be comparatively easier. Adaptive testing can provide a measure for students’ knowledge on an individual basis and is important to determine the level of understanding each student has of the subject material. Again, the benefit from computer-based assessment is that it is immediate. Adaptive testing is possible without computers, but computers can make it much more personalised and responsive.

Assessments can be conducted in a variety of ways. Assessment are conducted where students are required to provide a short answer or select a correct answer from provided alternatives. Alternatively, assessments can be conducted where students are required to formulate responses in the form of essays or perform a particular experiment in a lab. All of these assessments perform a vital role in measuring the students recall of subject material and cognitive thinking.

In an education environment, SAQs play a vital role for improving teaching (Süzen et al., 2020). More specifically, Larsen, Butler, and Roediger III (2008) and McDaniel et al. (2007) show that SAQs improve the retention capability of the learners. The authors show that students retain the material that they are tested on better than the material they are not tested on. This phenomenon is known as the testing effect. The authors also show that the testing effect is more prominent in SAQs than MCQs.

Further, Sychev, Anikin, and Prokudin (2020) point out that adding assessment to courses makes the education process more powerful by actively involving students in the education process. Through assessment, the students can assess whether they are actually learning, and then focus on their areas of weakness.

In addition to evaluating student learning, assessment can also provide feedback to guide future learning (Yang, Razo, and Persky, 2019). Feedback helps students realise their shortcomings and then promote students to improve on the shortcomings through learning (Siddiqi and Harrison, 2008).
With the advantages of SAQs and availability of easily scalable online courses, the grading of the student’s assessment puts a heavy workload on the teachers (Sychev, Anikin, and Prokudin, 2020). The delay on feedback can directly hamper students’ progress. In this context ASAG would allow students to attend a lecture, then do this sort of lightweight (and, to the institution, almost free) assessment. The quick turnaround of feedback then can help student improve learning.

In this chapter, I will look at how the literature approaches short answer questions and the computational systems that can automatically grade student responses to short answer questions.

I focus only on short answer question for which there is a definite answer, thus I exclude questions which require students to provide novel solutions, for example, *poetry*. Although, short answer question methods can be used as building blocks for grading essays (Taghipour and Ng, 2016), I will mostly discuss short answer questions in this dissertation.

I will also discuss in detail the representative work to date in ASAG, and consider some of the most successful approaches in the literature.

### 2.2 Methods for ASAG

Early work on ASAG relied on manually extracted templates such as regular expressions from expert-provided reference answers (Mitchell et al., 2002; Bachman et al., 2002). Such templates highlighted key concepts representing correct and incorrect answers.

The next category of work used *alignment* - matching words/phrases from a student’s answer with a given reference answer. The alignment approach then evolved into *machine learning* approaches. In machine learning approaches, many features are used along with alignment to grade student responses. The next step was to
use *deep learning methods* where features are learnt from the data and the learnt features are used to grade student responses.

In this chapter, I will first analyse systems that fall into each of these categories; templates, alignment, non-neural machine learning, and deep learning. With the knowledge from each of these systems, I will then discuss the methods that I will follow to tackle the problem of ASAG.

### 2.2.1 Use of Templates for ASAG

One of the earliest effective systems for ASAG is the AutoMark system (Mitchell et al., 2002). AutoMark grades student responses by comparing the student’s response with the mark scheme answer. Mark scheme answers are acceptable and unacceptable examples provided by the field experts/question setters. Each mark scheme answer is converted into a mark scheme template. Each template represents one form of an acceptable (or a specifically unacceptable) answer. To generate a mark scheme template, first, a mark scheme answer is pre-processed to standardise the input in terms of punctuation and spelling. Then, a sentence analyser identifies the main syntactic constituents of the text, and how they are related. A semantic component then generates synonyms for words in the mark scheme answer. An example mark scheme is shown in Figure 2.1 which illustrates a template for the mark scheme answer shown in Example 2.1.

**Example 2.1**

"The Earth rotates around the Sun"

Student responses are first parsed, and then matched against each mark scheme template, and a mark for each answer is computed. The representation of the templates is such that they can be robustly mapped to multiple variations in the input text.

With this template, a forward combination of any word (from 'start' to 'end') from
The performance of the system was tested in collaboration with the Centre for Research in Primary Science and Technology\(^1\). Statutory national curriculum assessment of science for pupils at age 11 was the domain explored. Test items were drawn from archive material. AutoMark achieved an accuracy of 92% testing on the 420 student responses. Out of the total responses tested, 315 of the 420 responses required a single word response and 105 of the 420 required forming explanatory sentences. The majority of the incorrect decisions by the system were on the responses requiring explanatory sentences. This was mainly caused by the inability of the syntactic parser used to correctly parse all the responses and also due to the mark

---

\(^1\)www.cripsat.org.uk
A similar concept of pattern matching is used in WebLAS (Bachman et al., 2002) to detect similar terms in the reference answer and the student’s response. The WebLAS system uses the Link Grammar Parser (Sleator and Temperley, 1995) to parse reference answers into chunks. The chunks are then analyzed by the human markers to provide appropriate weights. The weights help in determining partial and complete correctness of the response. WebLAS then uses WordNet (Miller, 1995) to collect similar words from the chunks. Finally, the chunks, their weights, and similar words are used to create regular expressions that can be matched with student responses.

The effectiveness of WebLAS was in detecting partial correct answers through weighted chunks. Identifying a partially correct answer is an important aspect of ASAG as this is a natural method of grading student responses.

A similar concept to WebLAS is also used in eMax (Sima et al., 2009). eMax requires the teacher to mark-up required semantic elements of the reference answers, accept or reject synonyms to these elements as prompted, and assign weights to each element for calculating the final score. The approach to grading is a combinatorial one, where all possible formulations are considered when pattern matching is performed. The assigned scores are also given a confidence rating so that difficult cases can be forwarded for manual review.

A major problem with systems that employ a mark scheme template is that it is difficult to predict the range of student responses without seeing an initial set of responses (Mitchell et al., 2002). Thus the template needs updating to accommodate for every variation of student response. As more variations of student response are collected, the amount of manual annotation increases, and the annotation task can eventually become infeasible. But having said that, unboundedness is a desirable property for
2.2. Methods for ASAG

an ASAG system. If a new variation (syntactic or semantic) to the answer is detected, it should be possible to grade response by simply updating the mark scheme for the particular answer/question. The system, however, should be robust enough to accommodate certain semantic or syntactic variations. For example, the system should be able to detect a response "Sun is orbited by Earth" for mark scheme in Example 2.1 without having to create a new mark scheme template. It is also evident that the mark scheme template can benefit from the addition of semantic comparisons of terms. For example, if the words like rotate and orbit can be recognised as related words then only one needs to be kept in the mark scheme.

In a step towards solving problem of infeasibility for manually updating mark scheme template, Willis (2015) developed the Amati system that automatically generates variations of template from seed templates. Amati marks student responses using a mark scheme template that can detect the presence of a term, a certain template (to capture spelling mistakes/variations), and if any terms precede others in a specified window. The author then uses Inductive Logic Programming (Lavrač and Džeroski, 1994) as a rule learner to automatically expand the seed mark scheme template created by using additional marked student responses. A training set and a test set of student responses were built from eight questions taken from an entry-level science module. The accuracy in the test set ranges from a maximum of 98.42% to a minimum of 91.0% on different questions. Amati however does not detect partially correct responses and is currently limited to binary classification of correct and incorrect responses.

As done in Willis (2015), eMax also uses a tool that helps the marker/teacher develop new mark scheme more easily. FreeText (Jordan and Mitchell, 2009) also provides a graphical user interface for teacher answer input. The templates are generated from the teacher’s answer using natural language processing. Through the interface, the teacher can specify mandatory keywords from the teacher answers and
select from synonyms provided by thesauri support. Both acceptable and unacceptable answers can be defined, and student responses are awarded credit according to template matches. A similar system, PMatch (Jordan, 2012) performs word-level pattern matching where all required words, word stems, and allowed synonyms for correct answers are matched by regular expressions against the teacher answers.

### 2.2.2 Alignment for ASAG

An alignment is the process of matching the words or phrases in a student response with the word or phrases in the answer provided by reference answer. In the field of ASAG, reference answers are the model answers provided by a subject expert (Heilman and Madnani, 2013; Dzikovska et al., 2013b; Pérez et al., 2005). Figure 2.2 shows an alignment between a student response and a reference answer.

Aligning words or phrases between a reference answer and the student response is one of the most popular methods for short answer grading. Alignment approaches have used words and combinations of words as units of alignment (Jordan, 2012; Siddiqi, Harrison, and Siddiqi, 2010; Cutrone, Chang, et al., 2011), or parse tree representations (Mitchell et al., 2002).

Auto-Assessor (Cutrone, Chang, et al., 2011) grades single-sentence student responses based on bag-of-words matching and synonyms with WordNet (Miller, 1995). The bag-of-words matching refers to matching individual terms between reference and student response. In Auto-Assessor, each word that matches exactly is given one
point, related words from WordNet are given partial points, and the rest are given no points. In addition to alignment, Auto-Assessor also performs preprocessing involving the removal of question words from answers and removal of stop words. The authors of Auto-Assessor do not report any empirical evaluation but mention that they get encouraging results. However, their system is constrained to well-formed, grammatically correct responses which is not always the case in the short answer grading task.

IndusMarker (Siddiqi and Harrison, 2008) aligns the parse representations of the reference answer and the student response instead of single words. First, both the sentences are parsed using the Stanford Parser (De Marneffe, MacCartney, Manning, et al., 2006). The parse representation is then processed into a Question Answer Language (QAL) (Siddiqi and Harrison, 2008). The QAL keeps track of word order and also similar words within sub-trees of the parse. The answer is then graded by matching QAL-encoded sub-trees between the reference answer and the student response. The authors of IndusMarker do not provide any details about the dataset. The claimed human-system agreement rate for IndusMaker is on average 98% for simple short-answer types like true/false, sentence completion, and single phrase generation. The claimed rate is an average of 90% for more complex short-answer like short explanations, concept definitions, and “compare and contrast” style questions. The structural alignment outperforms word level alignment baseline in every short-answer type.

Atenea (Pérez et al., 2005) uses both lexical and semantic alignment between the reference answer and the student response. BLEU (Papineni et al., 2002) is a technique that makes a weighted alignment between words or phrases between two sentences and outputs a value representing relatedness between two sentences. The core idea of using BLEU is that the more similar a student’s response is to the reference answers, the better it is, and, consequently, it will have a higher score. Latent Semantic Analysis (LSA) (Deerwester et al., 1990) is used for semantic matching. LSA uses
co-occurrence information obtained from a large corpora to form term-vectors. The
term-vectors are then used for similarity measurement. Atenea was evaluated on a
dataset collected from a real exams in Spanish where the responses were scored in the
range of $[0, 1]$. BLEU on its own achieved a correlation of 0.31, and LSA on its own
had 0.41 correlation with human scores. Combining both BLEU and LSA obtained
the best result of 0.50 correlation with human scores. This shows the importance of
both lexical and semantic information in ASAG.

The alignment problem depends on lexical matching and also semantic matching.
As seen in the above systems, different approaches are used to perform lexical and
semantic matching. In addition, as shown in Figure 2.2, an alignment system should
have knowledge of phrasal alignment, along with knowledge of contradiction.

Contemporary neural networks are effective tools for aligning texts (Bahdanau, Cho,
and Bengio, 2014). Such neural networks use resources like Word2Vec (Mikolov
et al., 2013a) which is built from a very large collection of data. Word2Vec gives a
semantic representation of words in the form of a vector of numbers. The related-
ness between two words then can be calculated easily by measuring the geometric
distance between vectors of two words. Neural networks are also effective in phrasal
alignment and in detecting contradiction (Parikh et al., 2016). Grading systems also
use a neural network for alignment and outperform strong non-neural baselines as
shown in automatic essay scoring task (Taghipour and Ng, 2016). The same model
used in Taghipour and Ng (2016) is ported for the short answer grading task by Rior-
dan et al. (2017). The results from Bahdanau, Cho, and Bengio (2014), Parikh et al.
(2016), Taghipour and Ng (2016), and Riordan et al. (2017) show that neural net-
works can effectively model alignment and produce strong results. Neural network
models are one of the most important aspects for us to investigate for the ASAG task.
I will discuss the neural alignment technique in more detail in Section 2.5.

Alignment has always been an effective and intuitive method for ASAG task, but
alignment in itself is not enough to get high accuracy in ASAG. However, alignment has always been a base from which many popular ASAG models are built (Dzikovska et al., 2013a; Ott et al., 2013; Sultan, Salazar, and Sumner, 2016).

2.2.3 Machine Learning for ASAG

Machine learning methods represent statistical models for ASAG compared to the information retrieval model of template matching. Machine learning models use a variety of features to learn a model from a given dataset. The goal of machine learning methods is to learn the concept of the correct answer for a question and use the knowledge to grade the student responses. In some sense, this task can be compared to measuring similarity between the student response and the reference answer.

Before detailing specific systems that use machine learning for automatic assessment, it is important to review methods of semantic similarity, as many methods for automatic grading use semantic similarity concepts (Mohler, Bunescu, and Mihalcea, 2011; Ramachandran, Cheng, and Foltz, 2015; Burrows, Gurevych, and Stein, 2015).

2.2.3.1 Background on Semantic Similarity

Early work in sentence similarity (Mihalcea, Corley, and Strapparava, 2006; Gabrilovich and Markovitch, 2007) involved identifying semantically similar word pairs (words with similar meaning) across sentences and then using the word pairs to measure the similarity between the sentences. These systems predominately used WordNet (Miller, 1995) as a semantic database to identify word-level semantic similarity. These systems correspond to the alignment methods for automatic grading.
Recent developments in the modelling of word meaning make more use of the *distributional hypothesis* of meaning, which states that the meaning of words show similarity if they occur in same context (Mikolov et al., 2013c). The distributional hypothesis allows meaning representations to be learnt from a corpus. Resources such as Word2Vec (Mikolov et al., 2013c) are built using the distributional hypothesis (Turney and Pantel, 2010), and are increasingly used in *Semantic Textual Similarity* (STS) tasks.

Additionally, the extension to distributional semantics models are contextualised models. Contextualised models produce representations of words based on the context they appear. A popular contextualised model is BERT (Devlin et al., 2018) which uses a Transformer architecture (Vaswani et al., 2017) to produce contextual embeddings (McCann et al., 2017). BERT-based models have been shown to be effective in extracting information from text, leading to significant improvements on many NLP tasks, including open-domain question answering, FAQ retrieval, and dialogue generation Wang et al., 2019; Sakata et al., 2019; Kim et al., 2020.

Semantic similarity can be defined in terms of commonality and differences of meaning between a pair of sentences. Commonality is defined as the intersection of the features of two distinct sentences, where these features can be lexical or semantic word-level representations (Lin, 1998). An example taken from Lin (1998), if A is an orange and B is an apple. The proposition that states the commonality between A and B is “fruit(A) and fruit(B)”. The more commonality that exists between a pair of sentences, the more similar they are; the more differences they have, the less similar they are (Lin, 1998). This definition leads to the estimation of similarity as a graded value which increases as commonality increases and decreases as difference increases. The grade can be a real valued number range showing semantic equivalence to semantic similarity. A semantic textual similarity task in SemEval 2012 (Agirre et al., 2012) defines such grades in terms of six levels where a score of 5 represents *semantic equivalence* and score of 0 indicates that *no relation* exists between
the sentences. Table 2.1 shows the similarity scores assigned by human annotators, along with the explanation of the scores provided by the task organisers. The performance of an STS system is measured by comparing with such gold annotated data.

<table>
<thead>
<tr>
<th>Score</th>
<th>Sentences</th>
</tr>
</thead>
</table>
| 5     | The two sentences are completely equivalent, as they mean the same thing.  
       | 1. Birdie is washing in the sink.  
       | 2. The Bird is bathing in the sink. |
| 4     | The two sentences are mostly equivalent, but some unimportant details differ.  
       | 1. In May 2010, the troops attempted to invade Kabul.  
       | 2. The US army invaded Kabul on May 7th last year, 2010. |
| 3     | The two sentences are roughly equivalent, but some important information differs/missing.  
       | 1. John said he is considered a witness but not a suspect.  
       | 2. “He is not a suspect anymore,” John said. |
| 2     | The two sentences are not equivalent, but share some details.  
       | 1. They flew out of the nest in groups.  
       | 2. They flew into the nest together. |
| 1     | The two sentences are not equivalent, but are on the same topic.  
       | 1. The woman is playing the violin.  
       | 2. The young lady enjoys listening to the guitar. |
| 0     | The two sentences are completely dissimilar.  
       | 1. John went horse back riding at dawn with a whole group of friends.  
       | 2. Sunrise at dawn is a magnificent view to take in if you wake up early enough for it. |

Table 2.1: Similarity scores with explanation (Agirre et al., 2016)

From the example in Table 2.1 we can see that there are various challenges associated with STS. The preliminary requirement for a STS system is to identify semantic similarity between different phrases like washing and bathing. Additionally, from the sentence pair with score 2, we can see that a high word overlap is not a necessary condition for high similarity score – interestingly, functional words like into (usually disregarded as an effective feature in many NLP problems) play a vital role in similarity calculation, as seen in Table 2.1 (score 2).

Interest in semantic similarity increased with the introduction of Agirre et al.’s task at SemEval-2012. From then onwards, most publications of STS have been in the
proceedings for SemEval tasks. Many systems submitted in the task use a combination of lexical, semantic, and knowledge base features in a machine learning context to achieve impressive results.

2.2.3.2 Semantic Textual Similarity systems

The features for Semantic Textual Similarity (STS) are used within various settings to measure sentence level similarity. Common feature groups include (1) shallow string similarity measures such as the longest common subsequence (Wu and Palmer, 1994), word and character n-gram overlap (Lesk, 1986), (2) semantic similarity measures derived from lexical similarity resources (such as WordNet (Miller, 1995)) and distributional measures (such as Latent Semantic Analysis (Landauer, Foltz, and Laham, 1998)), and (3) syntactic measures such as dependency overlap.

For instance, Przybyla et al. (2016) train a regression model on a combination of different features. Although they achieve a correlation of above 0.8 in the STS dataset, they suffer from non-generalisation of features, i.e. the performance drops considerably when in-domain training data with properties very similar to the test dataset is not available. This is evident when the scores drop when training and test data comes from a different source. Their method also suffers when two sentences are almost similar except for some critical information, as evident in the example provided by the authors: *What is the difference between Erebor and Moria?* and *What is the difference between splicing and superimposition?*, as we can see that except for the last two words, which create a different meaning, the rest of the words used are the same. We explore the advantage of identifying critical information in more detail in Chapter 4.

Similar systems trained on a Support Vector Regression model show that word level features that match the semantic similarity between tokens (such as WordNet and LSA) are useful metrics for measuring similarity (Liu et al., 2015). Gabrilovich and Markovitch (2007) argue that semantic similarity is narrowly defined, and thus
requires vast world knowledge to compute the score rather than just the context provided by the pair of texts. In this regard, the authors use concepts drawn from Wikipedia to measure semantic relatedness. On the dataset collected by the authors, the method achieves 0.72 correlation with the human annotated test set.

Gabrilovich and Markovitch’s system uses a vocabulary of words and presence of words as features, and thus lacks any order sensitive information (often called a “bag-of-words” model). For example, the system fails to detect the difference between “cats climb trees” and “trees climb cats”. The concept of order is important for STS and ASAG system. We use and compare both bag-of-words model and model that preserves word-order in Chapter 3 to investigate the importance of order for ASAG. Neural network architectures that can encode sequential information (Tai, Socher, and Manning, 2015a; Pagliardini, Gupta, and Jaggi, 2017; Shao, 2017) are used to measure text similarity and capture important sequential information. These neural network use the concept of Recurrent Neural Network (RNN) (Elman, 1990) and Long Short-Term Memory (Graves, Jaitly, and Mohamed, 2013) to capture sequential information of a text. More specifically, LSTM has internal mechanisms called gates that can regulate the flow of information. These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions. A more detail description of RNN and LSTM is provided in Section 2.5.3.2.

Alignment between words or phrases is also one of the important features for measuring similarity. It can be observed that systems that directly use alignment (Wu et al., 2017) between words (or phrases) for similarity calculation, or use alignment as one of the features (Maharjan et al., 2017; Tian et al., 2017), outperform other systems that do not use alignment (Bär et al., 2012; Han et al., 2013). This is a major motivation for us to use alignment for ASAG. We experiment with alignment approaches in Chapters 3, 4, and 5.
Aggregation is a technique for combining information from alignment. The combination can be either a summation, multiplication, or subtraction of the information from alignment. The combination can also be learned from the training data (Chang et al., 2010). A commonly employed technique for aggregation (Chang et al., 2010) is first to identify an alignment relation between sentence pairs, and to then use this relation to predict the overall score. Such methods suggest a pipeline, in which the first step in the alignment between sentence pairs is identified, then the alignment is used as a feature in a subsequent step to calculate similarity scores. In this regard, the alignment acts as an intermediate representation of sentences for calculating similarity.

The alignment mainly focuses on finding the best combination of pairs of similar semantic units in similar contexts. MacCartney, Galley, and Manning (2008) proposed a phrasal aligner that represents alignments as edit operations that are required to convert one sentence into another. Similar edit operations are also used in dependency parses of sentences to convert the parse tree of one sentence into another (Yao et al., 2013). Sultan, Bethard, and Sumner (2014) used a pipeline of alignment model whereby different aspects of sentences are aligned in sequential settings as shown in Figure 2.3. They also used contextual information provided by a dependency parse to improve alignment. They devised rules that aligned two different dependency types (rcmod, dobj) if they shared similar words in different sentences. For example, consider the sentences S1 and S2 in Figure 2.4.

In Figure 2.4, wrote and book have the typed dependency dobj(wrote, book) in S1, while book and wrote have the typed dependency rcmod(book, wrote) in S2. We
Figure 2.4: Example of alignment with dependency parse.

therefore get a rule that *dobj* can be aligned with *rcmod* as they both share a common word *wrote*. Such rules were learnt from annotated data, the dependency relation discovered was then used to align words.

Similarly, Tekumalla and Jat (2016) developed the iMatch algorithm for alignment. Tekumalla and Jat use Integer Linear Programming to align chunks from a pair of sentences. Their technique is to generate an alignment that maximises the overall similarity of the sentence pair. The similarity is calculated using both lexical and semantic features.

The neural alignment approach (Bahdanau, Cho, and Bengio, 2014; Parikh et al., 2016) has also shown promising results for alignment. Parikh et al. (2016) used the joint model where alignment and aggregation for similarity were trained jointly in a single neural network architecture. The approach outperforms more complex
neural network methods. The method is described in detail in Chapter 3. Although
the model shows promising results, it has the usual shortcomings of other neural
network approaches: it depends greatly on the amount of data available, it shows
poor performance on a small dataset, and it is hard to encode non-word features like
dependency parses.

Some of the important points from the above discussions are:

1. While measuring similarity, it is important to provide higher weights to critical
   information within the sentences.

2. Word level features are useful but must be followed with additional aggregation
   steps.

3. There is a need for more contextual information than is provided by the data.
   This can also be useful to identify critical information.

4. Structural alignment is better for STS task than word level alignment.

Since the ASAG task can also be modelled as the STS task, the above mentioned
points are also very important for the ASAG task. In this dissertation I will show
how these important points can be used to build accurate ASAG models. I use the
concept of critical information in Chapter 4. I use the concept of aggregation in
Chapter 3. I investigate the importance of structural alignment in Chapter 3.

2.2.3.3 Use of Similarity in Automatic Grading

The text similarity measures presented in section 2.2.3 are a popular feature for short
answer grading systems (Mohler, Bunescu, and Mihalcea, 2011; Ramachandran,
Cheng, and Foltz, 2015). The systems that use text similarity in automatic grading
measure semantic similarity between the reference answer and the student response.
If there is a high similarity between the reference answer and the student response,
then the response is marked as correct. The threshold for partial correctness is learnt
during the training of the similarity model.

The similarity features are combined with various machine learning techniques for
automatic grading. CAM (Content Assessment Module) (Bailey and Meurers, 2008)
uses a k-nearest neighbour classifier (Dudani, 1976) and features that measure the
percentage overlap of content with word overlap, chunk overlap, and nearest neigh-
bour similarity score. The overall similarity score was computed using Pointwise
Mutual Information (PMI) (Turney, 2002) between the reference answer and student
responses. The authors collected the data of 566 responses from 73 different ques-
tions. The system achieved an accuracy of 78% on the development set and 67%
on the test data for identifying student’s response as correct or incorrect. CAM is a
direct port of STS methods to the ASAG task. Although using semantic similarity is
an important method, the ASAG task also requires ASAG specific features as shown
by low accuracy using CAM. In this dissertation we will explore both STS methods
and also ASAG specific methods.

2.3 Datasets for ASAG

For other systems in automatic grading it is important to introduce a popular short
answer automatic grading dataset. The following sections introduce the dataset and
also analyse various systems that use the dataset.

Dataset Availability There are not many publicly available datasets for the ASAG
task. The primary reason behind this is that the many authors publishing in ASAG
domain are academics using datasets from their own university or other educational
institutions. Such datasets usually have confidentiality requirements which prevent
the author from releasing the full datasets. A few of the example systems using such
datasets are AutoMark, Amati, emax, and Freetext. The only datasets that I know of
that are publicly available are the ones connected to Mohler, Bunescu, and Mihalcea (2011) and Dzikovska et al. (2013a).

Referenced Based Dataset In the context of ASAG, it is the norm for several correct responses to be provided (Reynolds et al., 2010; Dzikovska et al., 2013a). We term this a referenced based dataset. In real answer marking contexts, multiple reference answers are provided to accommodate the potential variation in students’ correct responses. For example, the mark scheme for the UK GCSE (General Certificate of Secondary Education) Computer Science exam gives multiple proposed answers for free-text response questions, as shown in Example 2.2.

Example 2.2

question: Why do the schools need wired network?

reference answer 1: as they need high bandwidth / reliable systems as delays in lessons are not acceptable

reference answer 2: network performance may be poor if there are large numbers of student-owned devices using the school network at any one time

The other form of dataset is one without reference answers. Such datasets do not have reference answers and only consist of questions, and student responses, with the marker typically being expected to be able to judge whether the student responses are correct. A subset of the student responses are graded manually for training purpose. An example of such a dataset is the Automated Student Grading Prize - Short Answer

\[\text{https://filestore.aqa.org.uk/resources/computing/AQA-85202-SMS.PDF}\]
2.3. Datasets for ASAG

Scoring (ASAP-SAS) dataset\textsuperscript{3}. As this dataset is no longer available, we do not perform any experiments with it.

The ASAG dataset used in this thesis is a referenced based dataset that contains samples from the Beetle (Dzikovska, Nielsen, and Brew, 2012) and SciEntsBank (Nielsen et al., 2008) corpora. The dataset was introduced in SemEval challenge (Dzikovska et al., 2013a). We will term this dataset as the SemEval dataset. The Beetle dataset consists of electricity and electronics questions and the SciEntsBank dataset contains questions from different science domains for 3rd to 6th-grade students. The Beetle dataset is mainly comprised of 56 questions in the domain of basic electricity and electronics requiring one or two sentence answers, and it has nearly 3000 student answers to those 56 questions. The SciEntsBank dataset contains approximately 10,000 answers to 197 assessment questions in 15 different science domains. The dataset consists of questions, along with a set of reference answers provided by a domain expert. The student response are categorised into either the 2-way labels into

1. correct or

2. incorrect

or the 5-way labels, a more fine-grained classification of responses into the categories

1. correct,

2. partially correct incomplete,

3. contradictory,

4. irrelevant, and

5. non domain.

\textsuperscript{3}https://www.kaggle.com/c/asap-sas/overview
Chapter 2. Background

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Train</th>
<th>UA</th>
<th>UQ</th>
<th>UD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beetle</td>
<td>3941</td>
<td>819</td>
<td>439</td>
<td>-</td>
</tr>
<tr>
<td>SciEntsBank</td>
<td>4969</td>
<td>540</td>
<td>733</td>
<td>4562</td>
</tr>
</tbody>
</table>

Table 2.2: Number of training cases, unseen answers, unseen questions, and unseen domains in the corpus.

<table>
<thead>
<tr>
<th>Label</th>
<th>Train (%)</th>
<th>UA</th>
<th>UQ</th>
<th>Test-Total (%)</th>
<th>Train (%)</th>
<th>UA</th>
<th>UQ</th>
<th>UD</th>
<th>Test-Total (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>correct</td>
<td>1665 (0.42)</td>
<td>176</td>
<td>344</td>
<td>520 (0.41)</td>
<td>2008 (0.40)</td>
<td>233</td>
<td>301</td>
<td>1917</td>
<td>2451 (0.42)</td>
</tr>
<tr>
<td>partially correct</td>
<td>919 (0.23)</td>
<td>112</td>
<td>172</td>
<td>284 (0.23)</td>
<td>1324 (0.27)</td>
<td>113</td>
<td>175</td>
<td>986</td>
<td>1274 (0.22)</td>
</tr>
<tr>
<td>contradictory</td>
<td>1049 (0.27)</td>
<td>111</td>
<td>244</td>
<td>355 (0.28)</td>
<td>499 (0.10)</td>
<td>58</td>
<td>64</td>
<td>417</td>
<td>539 (0.09)</td>
</tr>
<tr>
<td>irrelevant</td>
<td>113 (0.03)</td>
<td>17</td>
<td>19</td>
<td>36 (0.03)</td>
<td>11115 (0.22)</td>
<td>133</td>
<td>193</td>
<td>1222</td>
<td>1548 (0.27)</td>
</tr>
<tr>
<td>non-domain</td>
<td>195 (0.05)</td>
<td>23</td>
<td>40</td>
<td>63 (0.05)</td>
<td>23 (0.005)</td>
<td>3</td>
<td>0</td>
<td>20</td>
<td>23 (0.004)</td>
</tr>
</tbody>
</table>

Table 2.3: Label distribution. Percentages in parentheses. UA, UQ, UD correspond to individual test sets.

The Beetle dataset consists of a training set and two distinct test sets: unseen answers (UA) and unseen questions (UQ). The SciEntsBank dataset also contains an Unseen domains (UD) test set.

**Unseen Answers** (UA) is a test set to assess system performance on the responses to questions contained in the training set for which the system has seen example student responses.

**Unseen Questions** (UQ) is a test set to assess system performance on responses to questions for which the system has *not* previously seen the questions, but which still fall within the question domains represented in the training data.

**Unseen Domains** (UD) is a test set to assess system performance on responses to questions for which the system has *not* previously seen example student responses on particular topics. This test set measures the domain adaptability of a system.

The final label distribution for training and test data is shown in Table 2.3.

Following are some of the examples of questions, reference answers, and students’ responses taken from the Beetle and the SciEntsBank dataset.

**Example 2.3**

*Beetle (UA)*
question: When switch X was closed and switch Y was open, why was bulb A on?

reference answer 1: Bulb A was still contained in the same closed path with the battery.

reference answer 2: Bulb A is still contained in a closed path with the battery and switch X.

Student #1 (C): because it was contained in a closed path to the battery.

Student #2 (P): Because it’s path was closed.

Student #3 (I): there was a gap in there

Example 2.4

Beetle (UQ)

question: How does a damaged bulb compare to an open switch?

reference answer 1: A damaged bulb and an open switch both create a gap and cause bulbs to go out.

reference answer 2: there is a gap in the circuit

Student #1 (C): A damaged bulb and open switch created gaps in the path.

Student #2 (P): it creates a gap.
Student #3 (I): They are similar.

Example 2.5

SciEntsBank (UA)

question: You used several methods to separate and identify the substances in mock rocks. How did you separate the salt from the water?

reference answer: The water was evaporated, leaving the salt.

Student #1 (C): We evaporated the water.

Student #2 (P): Get all the water out.

Student #3 (I): We kept the salt in the vial and put the water in the Petri dish.

Example 2.6

SciEntsBank (UQ)

question: What happens to earth materials during erosion?

reference answer: Earth materials are worn away and moved during erosion.

Student #1 (C): During erosion earth materials break away and flow away.

Student #2 (P): They just move around.
2.3. Datasets for ASAG

Student #3 (I): The materials rise and disappear.

Example 2.7

SciEntsBank (UD)

question: Pam sprayed water on her mother’s car. The water drops looked just like those she had seen on other surfaces that do not absorb water. What property of water causes the drop to be shaped that way?

reference answer: Surface tension causes the drop to form a dome or bead shape.

Student #1 (C): The surface tension. Surface tension is holding it up in a bead.

Student #2 (P): The property of tension.

Student #3 (I): There was no surface tension.

2.3.1 Evaluation

The evaluation of the SemEval dataset is performed using per-class $F_1$ score. The summary $F_1$, including all the classes is evaluated as either:

1. Microaverage is the average value of $F_1$ across classes. This metric favours systems that perform well across all classes, regardless of class size.

2. Weighted Average is the value of $F_1$ weighted by class size. This metric favours systems that perform best on the class label with largest number of examples.
Weighted average prefers systems that perform best on the largest number of examples, favouring higher performance on the most frequent classes. However, Dzikovska et al. (2013a) reported that in practice, only a small number of the systems were ranked differently by the different metrics. The majority of the rankings were similar for both the metrics. Because of this report, in this thesis we will only use weighted average to report the performance of the systems.

The author of the SemEval dataset also presented a baseline system as a part of an ASAG challenge (Dzikovska et al., 2013a). The baseline system for the dataset used a lexical similarity score computed using word overlap, Lesk score, F1 score and cosine similarity. These four features were trained using a C4.5 decision tree (Ho, 1998).

One of the top performing system in the challenge is SoftCardinality (Jimenez, Becerra, and Gelbukh, 2013). SoftCardinality is defined in terms of similarity between words of the sentence and a collection of elements. Softcardinality (S) is calculated with the following expression:

\[
|S| = \sum_{i=1}^{n} w_i \left( \sum_{j=1}^{n} \text{sim}(s_i, s_j)^p \right)^{-1}
\]  

having:

- \( S = \{s_1, s_2, ..., s_n\} \),
- \( w_i \geq 0 \),
- \( p \geq 0 \),
- \( 1 > \text{sim}(x, y) \geq 0 \),
- \( x \neq y \), and
- \( \text{sim}(x, x) = 1 \).
The parameter $p$ controls the degree of "softness" of the cardinality (the larger the "harder"). In fact, as $p$ becomes arbitrarily large ($p \to \infty$), the softcardinality is equivalent to classical set cardinality. The default value for this parameter is $p = 1$. The coefficients $w_i$ are weights associated with each element, which can represent the importance or informative character of each element. The function $\text{sim}$ is a similarity function that compares pairs of elements in the collection $S$. For example if we take a dense vector representation of words as follows:

- “hotel” = [0.286 0.792 -0.177 -0.107 0.109 -0.542 0.349 0.271]
- “motel” = [0.280 0.772 -0.171 -0.107 0.109 -0.542 0.349 0.271]

and define the $\text{sim}(x, y)$ function as a cosine similarity function between the dense vector. With a uniform weight of 1 and $p = 1$, the SoftCardinality between “hotel” and “motel” is calculated from Equation 2.1 as:

$$S = \left(\frac{1}{\text{sim}(\text{hotel}, \text{motel})}\right)^{1/p}$$

or, $S = \frac{1}{0.99}$

or, $S = 1.09$

The equation 2.1 gives a semantic representation of a sentence. This representation is then used to provide values for the question, collection of reference answers, student response, the intersection of reference answer and student response, and the intersection of question and student response. These features are then used in a classification algorithm to predict labels for student responses. The similarity value for words is calculated using a similarity measure, with $w_i$ set to 1.

I believe there is a scope in using the variable $w_i$ because the importance of each word in the response is different. Some words can be critical for determining correctness of answers. I explore this avenue further in Chapter 4.
Similarly ETS (Heilman and Madnani, 2013) used all of Dzikovska et al.’s baseline features (word overlap, Lesk score, F1 score and cosine similarity) along with edit distance features. Edit distance was measured as the number of characters which needed to be changed in the reference answer to obtain the student response. Additionally, Heilman and Madnani also used a domain adaptation technique. The domain adaptation technique used the concept that only general features can be ported from one domain to another and domain specific features should be left out. For this, the authors created three sets of combinations of different features and trained three different models. The first set termed ETS-1 uses all the baseline features and text similarity features using BLEU (Papineni et al., 2002) and edit distance. The second set termed ETS-2 uses all the baseline features and word-overlap feature. It does not use any external resources other than the provided data. The third set termed ETS-3 is the combination of features from ETS-1 and ETS-2. The authors concluded that the semantic similarity features were better for domain adaptability than lexical semantic features like word/character overlap.

A recent work which involved combining various STS features (see section 2.2.3.1) improved results on the SemEval dataset (Sahu and Bhowmick, 2019). Sahu and Bhowmick used semantic similarity features like knowledge-based measures, corpus-based measures, and word embedding features. They also used word-overlap feature and LDA. All of these features are explained in detail in section 2.2.3.1. Word embedding features are word vectors that contain semantic information of words (see section 2.5.2). The authors also introduced relevance feedback-based features. Relevance feedback-based features are based on the idea that the provided reference answer does not cover all the appropriate concepts and the semantic variations that are required to answer a given question. Thus, it is appropriate to add all the correct student responses that have high similarity to reference answers, and also student responses that have low similarity with reference answers. These features were used in various models for ensemble learning. Ensemble learning (Mendes-Moreira et al.,
Table 2.4: Weighted average F1 scores for different systems in the SemEval challenge (Dzikovska et al., 2013a; Sahu and Bhowmick, 2019)

<table>
<thead>
<tr>
<th>System</th>
<th>Beetle ua</th>
<th>uq</th>
<th>SciEntsBank ua</th>
<th>uq</th>
<th>ud</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>0.483</td>
<td>0.463</td>
<td>0.435</td>
<td>0.402</td>
<td>0.396</td>
</tr>
<tr>
<td>SoftCardinality</td>
<td>0.558</td>
<td>0.450</td>
<td>0.537</td>
<td>0.492</td>
<td>0.471</td>
</tr>
<tr>
<td>ETS-2</td>
<td>0.705</td>
<td>0.614</td>
<td>0.625</td>
<td>0.556</td>
<td>0.434</td>
</tr>
<tr>
<td>Feature Engineering</td>
<td><strong>0.709</strong></td>
<td><strong>0.6248</strong></td>
<td><strong>0.925</strong></td>
<td><strong>0.658</strong></td>
<td><strong>0.656</strong></td>
</tr>
</tbody>
</table>

2012) refers to the process of training multiple models and using a combination of such models to make predictions. The advantage of using ensemble learning is that it minimises the risk of using only a single poorly performing individual model. The various models that were used in the ensemble are Tree, Bagging tree, Boosting tree, Support vector regression, Linear regression, LASSO, ElasticNet, Kernel ridge regression. Finally, Stacked Regression (Wolpert, 1992) is used to combine the results of each of the models. We term this approach Feature Engineering. The authors provide the result for only 5-way classification. We can see in Table 2.4 that the Feature Engineering approach outperforms all the other approaches.

One disadvantage of the Feature Engineering approach is that it depends on the accuracy of the tools used for feature extraction. Additionally, the feature extraction process is also computationally expensive. In this dissertation, I will investigate neural network approaches that do not require manual feature extraction and instead learn the features from the data. The detail on the neural network approaches is given in section 2.5.

Table 2.4 shows the system performance using weighted average F1 scores for 5-way classification on the dataset. We can see from the table that the ETS-2 method of using lexical features has a good performance. Interestingly, SoftCardinality performed better in the domain adaptability. Domain adaptability depends on the similarity features and since both the systems used different similarity measures, domain
adaptability depends upon which similarity measure has better performance. However, the feature engineering approach which combines a multitude of both lexical and semantic features performs better than all the other systems.

2.4 Summary of each ASAG Technique

One major issue with the ASAG task is unavailability of a variety of datasets. The work on the publicly available SemEval dataset has become a bit stagnant. ETS (Heilman and Madnani, 2013), domain adaptation (Heilman and Madnani, 2013), and SoftCardinality (Jimenez, Becerra, and Gelbukh, 2013) had state-of-the-art results for quite a long time on the SemEval dataset. Only recently has the Feature Engineering approach (Sahu and Bhowmick, 2019) outperformed all the other models by using a large set of features and training an ensemble of classifiers. The long gap in improving the state-of-the-art suggests that the ASAG task could benefit from using more successful and current neural network approaches.

It is evident from the techniques using templates that they are useful to accommodate new answers. The templates can be easily updated to grade new answers or questions. The alignment systems show that an accurate alignment paves the way for effective grading systems. While the alignment by itself may not be sufficient for grading students’ answers with high accuracy, it certainly can be used as a base from which to build a more robust grading system. The semantic similarity methods use the latest advances in NLP for ASAG, and pave the way for jointly learning the templates, alignment, and aggregation of the alignment for grading answers. The only issue of concern is that the semantic similarity task is not an exact replica of the ASAG task: a high degree of similarity does not mean that the answer is correct, nor does a low degree mean the answer is incorrect. The ASAG task is more defined with reference to a real-world task than is the semantic similarity task. For the ASAG task the relatedness between two sentences is defined with reference to the reality of what
the educators are trying to teach. So in comparing similarity, the markers have an understanding of what the key points are, whereas for the general semantic similarity task, that is less well defined.

There are ASAG systems that uses ASAG specific properties in addition to semantic similarity properties. Some examples of ASAG specific properties are removing words from question before making a comparison (Sultan, Salazar, and Sumner, 2016), manually weighting chunks to detect partial correctness (Bachman et al., 2002), or domain adapting models (Heilman and Madnani, 2013). To this end I will focus my attention towards improving the alignment and improving the weighing mechanism of chunks, improving domain adaptability to improve performance on the ASAG task.

An important aspect of ASAG system is also to identify critical information in the answer. The critical information is used in literature as focus (Ziai and Meurers, 2014) and Minimal Meaningful Propositions (MMP) (Godea, Bulgarov, and Nielsen, 2016). Focus is defined as a part of the answer that is a direct answer to the question under discussion. The following is an example taken from Ziai and Meurers (2014).

**Example 2.8**

**question**: Where did Schorlemmer go after the rally?

**reference answer**: He happily walked home

In Example 2.8 “home” is termed as focus. A focus needs to be a direct answer to the question, there is a higher restriction in the selection of focus. For example, for a question “where does she live?” and the answer “She lives in Berlin”, the focus
has to be “in Berlin” and not just “Berlin” as a phrase “near Berlin” would be incorrect answer to the question. The author present that the focus annotation of reading comprehension questions had a Kappa value ranging from 0.61 to 0.69. The agreement between the annotator shows that it is possible to extract critical information from answers. The agreement of the annotation could be improved by relaxing the strict criteria of focus by extracting information that is not the direct answer to the question but an essential part of the question. For example, we could extract only “Berlin” as important information and combine it with alignment methods to differentiate between “near” and “in”. We illustrate this property in depth in Chapter 4.

Similar to focus, minimal meaningful propositions is also a part of an answer that is a direct answer to the question. However, MMP can

1. primary: fundamental to answering question

2. secondary: relevant but not integral to answer – often clarify or qualify a primary MMP

3. extraneous: unnecessary or minimally relevant to the question

4. redundant: contain information directly or indirectly provided by the question

The following is an example taken from Godea, Bulgarov, and Nielsen (2016).

Example 2.9

question: How did Rutherford figure out that atoms are mostly empty space, and that the nucleus is positive?

reference answer: He used gold foil hammered about an atom thick, and placed radium in a lead lined box that emitted positive alpha particles
towards the gold foil.

**MMP #1:** Rutherford used gold foil with the thickness of an atom.

**MMP #2:** Rutherford placed radium in a lead lined box.

**MMP #3:** The lead lined box emitted positive alpha particles towards the gold foil.

Godea, Bulgarov, and Nielsen (2016) automatically extracted MMPs by first learning frequently used syntactic patterns from human-annotated MMPs. In the next phase, MMPs is from reference answer and student response by extracting the longest matching syntactic patterns from the answer and the response. The top three most frequently used syntactic structures are:

1. NP, VP, NP
2. NP, VP, NP, PP, NP
3. NP, VP, PP, NP

The algorithm extracted MMPs with an F1-score of 0.62 compared to an F1-score of 0.44 for a baseline that extracts a whole sentence as MMP. Although MMP can be useful for ASAG tasks, similar to focus it also has strict criteria of representing a complete answer. A better alternative will be to identify important information and then use it with a variety of other methods like alignment and semantic comparisons for ASAG tasks. We take the approach of extracting important information and then using it in an alignment setting in Chapter 4.
Recently, artificial neural networks have become very popular and effective in many semantic matching tasks such as semantic similarity, textual entailment, and question answering. Neural network models claim the state-of-the-art in most datasets in these fields. In this dissertation I will investigate the effectiveness of neural networks for the ASAG task.

An advantage of using a neural network model is that the network can automatically learn a configuration with weights values to create units for alignments (LeCun, Bengio, and Hinton, 2015). The alignments can then be matched to measure semantic similarity. The semantic information between matched pairs are then aggregated for the final classification. All of these steps can be represented in different layers of a neural network. The model can learn alignment in one layer, comparisons of the alignments in the next, and the classification of the comparisons in another. This can be achieved with minimum representation at input - usually an off-the-self embedding (Pennington, Socher, and Manning, 2014). This means that for a variety of tasks, manual annotation of features is not required in neural networks. Different forms of neural network can be used for a variety of tasks. For example, an attention neural model (Parikh et al., 2016; Kim et al., 2017) can automatically learn features directly from data and then align, compare, and aggregate them to grade student responses. In contrast, Ott et al.’s (2013) system uses various techniques and resources such as WordNet (Miller, 1995), mutual information (Turney, 2002), parsing and tokenisation to extract features. The errors in extracting these features are propagated in the pipeline methods of automatic grading.

2.5 Deep Learning Methods

Conventional machine-learning techniques require domain experts to build feature extractors that transform raw data into a representation or feature vector that the machine learning system (usually classifiers) can use as input.
Deep learning is a new-era in machine learning that characterises representation learning. Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification (LeCun, Bengio, and Hinton, 2015). Deep learning methods have multiple level of modules that transform the representation of data from raw input at the first layer to necessary representation at each successive layer for solving the given problem. In doing so, each module also suppresses any irrelevant information from the data. For instance, for calculating similarity between two sentences, a deep learning method’s first layer converts both the sentences into a semantic vectors. The second layer then matches these vectors to align words between the sentences. The third layer then compares the alignment to compute similarity/dissimilarity between words. Finally, the third layer then aggregates all of the information and makes the final classification. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

A deep learning network is a neural network with multiple processing layers between input neurons and output neurons. The basis for each of the processing layers is a feed-forward neural network.

Feed forward neural networks are a class of neural network architecture that do not contain any loops or cycles. An example is shown in Figure 2.5. In Figure 2.5, the leftmost layer of the network is called the input layer, and the rightmost layer the output layer. The middle layer of nodes is called the hidden layer, because its values are not observed in the training set. The circles labelled “+1” are called bias units. The network has parameters

\[(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})\]

where \(W_{ij}^{(l)}\) is the weight associated with the connection between unit \(j\) in the layer
\( l \), and unit \( i \) in the layer \((l+1)\). Thus, \( W^{(1)} \) is a real valued matrix with 3 rows and 3 columns and \( W^{(2)} \) is a real valued matrix with 1 row and 3 columns. The output value (activation) of unit \( i \) in layer \( l \) is denoted by \( a^{(l)i} \). Thus, for the given parameters \( W, b \), the neural network outputs a hypothesis \( h_{W,b}(x) \). The computation of the hypothesis is forward propagated in the following manner:

\[
    a_1^{(2)} = f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)})
\]

\[
    a_2^{(2)} = f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)})
\]

\[
    a_3^{(2)} = f(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)})
\]

\[
    h_{W,b}(x) = f(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)})
\]

Here the function \( f \) is an activation function that introduces non-linearity in the network. A non-linear activation function allows the network to predict the class from the data that is separated by a non-linear decision boundary. Commonly used activation functions are \textit{tanh}, \textit{sigmoid}, and \textit{ReLU}. The sigmoid function outputs a value between 0 and 1. Therefore, it is especially used for models where we have to
predict the probability as an output. The \textit{tanh} function outputs a value between -1 and 1. The mean of the activations that come out of the \textit{tanh} functions are closer to having a zero mean. As a result, data is more centered which makes the learning for the next layer easier and faster. ReLU is a function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

The advantage of using feed forward neural network is that we can swiftly calculate the output using matrix-vector representation and linear algebra. The network can have any number of hidden layers (deep-network) and any number of output neurons. The forward propagation in any layer can be calculated similarly as described above.

### 2.5.1 Softmax

Softmax (Goodfellow, Bengio, and Courville, 2018) normalises a vector of arbitrary values to a probability distribution between 0 and 1. The sum of the softmax output is equal to 1. Therefore, it is commonly used in the last layer of a neural network to predict probabilities of the possible output classes. The following is the mathematical expression for the softmax function for a vector $x$ of size $i$:

$$\text{Softmax}(x)_i = \frac{\sum e^x}{\sum_{i=1}^{K} e^x}$$

Here $K$ represents the number of classes. For example, for a classification problem with 3 classes we can use 3 nodes at the output layer. The softmax at the output layer will then show the probability of each class corresponding to a neuron.

Deep learning methods use multiple processing layers to learn various related representations of data, and have produced state-of-the-art results in many domains. Deep learning architectures and algorithms have already made impressive advances
in fields such as computer vision and pattern recognition. Following this trend, recent NLP research is increasingly focusing on the use of deep learning methods.

Earlier work on machine learning approaches for NLP problems mostly used models trained with statistical techniques such as Support Vector Machines, logistic regression and linear optimisation. These models are trained on manually curated sparse feature sets and lack multiple layers of adaptive features. Shallow models are effective in solving simple or well-constrained problems, but their limited modelling power can fail to generalise over variations present in real usage scenarios. In the last decade, neural networks based on dense vector representations have been producing superior results on various NLP tasks. The success is mostly due to use of word embeddings (Mikolov, Le, and Sutskever, 2013) and use of deep layers (Yu et al., 2011) to represent complex data. The deep (multiple) layers provide mechanisms to represent data. In contrast, NLP systems based on SVM, logistic regression or linear optimisation rely heavily on hand-crafted features. Such hand-crafted features are time-consuming to produce and often incomplete.

Collobert et al. (2011) demonstrated that a simple deep learning framework outperformed the then state-of-the-art approaches in several NLP tasks such as named-entity recognition (NER), semantic role labelling (SRL), and POS tagging. Following the success we also use deep learning methods to obtain state-of-the-art result in the ASAG task (see Chapters 3, 4 and 5).

The success of deep learning methods for NLP depends on their ability to represent words in the form of distributional vectors or word embeddings. Word embeddings follow the distributional hypothesis. The distributional hypothesis states that the words with similar meanings tend to occur in similar contexts (Harris, 1954). A word embedding is a vector representation of words where the representation is derived utilising the distributional semantics of words. In distributional semantics, the meaning of the words is derived from word usage. Although Caliskan, Bryson, and
Narayanan (2017) argue that distributional models mirror the data they are fed and thus can sometimes be subject to biases in the underlying data, this has proved an effective assumption in many areas of empirical NLP (Almeida and Xexéo, 2019). Some famous word embedding methods are Word2vec (Mikolov, Le, and Sutskever, 2013) and GloVe (Pennington, Socher, and Manning, 2014). (Pennington, Socher, and Manning, 2014).

### 2.5.2 Word Embeddings

Vector representations of words allow for a representation of semantically similar words. Word vectors that are related to one another are mapped onto points that are close to each other in a high dimensional space. Word2vec and GloVe are such models, trying to directly predict a word by using its neighbours, learning small but dense vectors called embeddings.

**Word2vec** is a computationally efficient, unsupervised model that learns word embeddings from raw text. In order to learn these dense vectors, Word2vec is available in two flavours: the CBOW model and the skip-gram model (Mikolov, Le, and Sutskever, 2013).

Word2vec is a three-layer neural network, where the first and last layers form the input and output; the intermediate layer builds latent representations for the input words to be transformed into the output vector representation. The Word2vec representation of words allows for exploring interesting mathematical relationships between word vectors, which is also an intuitive expression for words. For instance, we will can find out the value of the following expression by using Word2vec representation of words (Mikolov, Yih, and Zweig, 2013):

\[
\text{king} - \text{man} \approx \text{queen} - \text{woman}.
\]

Mathematically, what this expression evaluates is the equivalence of the latent space
of the word vectors evaluated by the expressions. On the other hand, intuitively, we can understand that removing man from king and adding woman results in queen. Such a relationship can be built only when the contexts of the words are understood, which is possible when the positional relationships of the words are exploited. It is evident, from the semantics, that the word king occurs in a position along with man, in a manner similar to how the word queen and woman are present with one another.

GloVe (Pennington, Socher, and Manning, 2014) is used to obtain vector representations of words using an unsupervised algorithm. The GloVe model is trained on the non-zero entries of a global word-word co-occurrence matrix. The word-word co-occurrence matrix shows how frequently words co-occur with one another in a given corpus. The main intuition behind the GloVe model is that the ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. The following example, presented in the original paper, explains this concept quite clearly: if we look at the words “steam” and “ice”, they have in common that they are both states of water, but they have different characteristics. So we might expect them to occur more or less the same amount of times around the word “water”, whereas “steam” will occur more often together with “gas”, and “ice” will occur together with “solid”.

The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words’ probability of co-occurrence.

Kenter and De Rijke (2015) successfully used word embeddings to estimate text similarity. Kenter and De Rijke (2015) used features like alignment, where the alignment features were the similarity values generated by measuring distance between the embeddings of the pair of words. Figure 2.6, presented in the original paper, depicts the alignment between words of a pair of text.

As can be observed from the figure, the two texts have terms that are close to each
2.5. Deep Learning Methods

**Figure 2.6**: (Kenter and De Rijke, 2015) Hypothetical example — two-dimensional representation of the word embeddings for two short texts (each consisting of three terms), represented as transparent and opaque dots respectively. The corresponding means of the two sets of embeddings are depicted as crossed circle

other (at the top and bottom in the figure), while the ones at the far left and right have no matching terms. Regardless of this discrepancy, the means of the two are close to one another. The authors indicate that a classifier can benefit from more elaborate information about the alignment of word embeddings across the semantic space, in contrast to just providing the mean similarity. The alignment feature is obtained by calculating the cosine similarity between word pairs from two texts and for each word from the first text taking the word pair with maximum cosine similarity value. The alignment between term \( w \) with respect to short text \( s \) is represented by \( \text{sem}(w, s) \):

\[
\text{sem}(w, s) = \max_{w' \in s} f_{\text{sem}}(w, w')
\]  

(2.2)

In Equation 2.2, the function \( f_{\text{sem}}(w, w') \) returns a cosine similarity between two words. For example if we take a dense vector representation of words as follows:

- “hotel” = [0.286 0.792 -0.177 -0.107 0.109 -0.542 0.349 0.271]
- “motel” = [0.280 0.772 -0.171 -0.107 0.109 -0.542 0.349 0.271]
- “shop” = [0.380 0.272 -0.171 -0.302 0.124 -0.452 0.349 0.101]

then using Equation 2.2, “hotel” aligns with “motel” and not with “shop” in this
context as the cosine similarity between “hotel” and “motel” is 0.99 and between “motel” and “shop” is 0.85.

The features are trained with a Support Vector Machine (SVM). Features generated using word embedding clearly outperformed other non-neural corpus based word representations such as methods using Pointwise Mutual Information (PMI) (Islam and Inkpen, 2008). To analyse the effect of each feature set, the authors perform an ablation study, where they leave out a feature and train the system again. The ablation study showed that there is a major dip in accuracy when the alignment feature is removed. Additionally, the ablation study also showed that removing the mean similarity feature has negligible effect on the accuracy. This shows the importance of alignment for text matching task and indicates that aligning words is a successful strategy for determining semantic similarity between texts. However, the caveat in this work was that alignment was pre-calculated as a feature set for the classification model. The obvious improvement to this approach is to use the representational power of the deep learning method to learn the alignment in one of the layer of the deep network.

**Contextual Embeddings**: Word embeddings like Word2Vec and GloVe only obtain a single global representation for each word. However, words can have different meanings with respect to their contexts. Contextual embeddings move beyond word-level semantics in that each token is associated with a representation that is a function of the entire input sequence. These context-dependent representations can capture many syntactic and semantic properties of words under diverse linguistic contexts (Devlin et al., 2018; Peters et al., 2018). Contextual Embeddings are usually trained via Language Modelling. A language model is a probability distribution over a sequence of tokens. A language model seeks to compute the probability of a word occurring in a text, given some prior history of words seen. Such models allow us to determine that if the phrase “I am going to write with a” is seen, then the word
“pencil” is more likely to be the next word than “frog”. Popular contextual embeddings are ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018). ELMo uses a 2-layer bidirectional LSTM (biLM) (Graves, Jaitly, and Mohamed, 2013) which is pre-trained on a large text corpus, to learn both words (e.g., syntax and semantics) and linguistic context (for example, to model polysemy). BiLMs capture context-dependent aspects of word meaning. BERT stands for Bidirectional Encoder Representations from Transformers. BERT proposes a masked language modelling (MLM) objective, where some of the tokens of a input sequence are randomly masked, and the objective is to predict these masked positions taking the corrupted sequence as input. BERT applies a Transformer encoder to attend to bi-directional contexts during pre-training. In addition, BERT uses a next-sentence-prediction (NSP) objective. Given two input sentences, NSP predicts whether the second sentence is the actual next sentence of the first sentence. The NSP objective aims to improve the tasks, such as question answering and natural language inference, which require reasoning over sentence pairs.

Both ELMo and BERT representations can be easily added to existing models and have been shown to significantly improve the state-of-the-art across challenging NLP problems, including question answering, textual entailment and sentiment analysis (Wang et al., 2019; Sakata et al., 2019; Kim et al., 2020).

### 2.5.3 Deep Learning for Text Similarity and Automatic Grading

The deep learning methods for text similarity and ASAG focus on creating a single representation for two sentences that can be used for classification. This is shown in Figure 2.7. The goal of these representations is to encode deeper sentence comprehension that can aid in measuring text similarity. The layers in these deep models use various neural network architectures like Convolution Neural Network (CNN) (LeCun, Bengio, and Hinton, 2015), Recurrent Neural Network (RNN) (Elman, 1990) or Attention (Bahdanau, Cho, and Bengio, 2014).
2.5.3.1 Convolution Neural Networks

A Convolution Neural Network (CNN) is a feed-forward neural network where each neuron in a layer receives input from a group of adjacent neurons in the previous layer. This way CNNs can learn multiple local level (lower level) features, which are later composed into higher level features. For example, in computer vision, CNNs can detect edges (higher level feature) in an image from pixels (lower level features), these edges can then be combined to detect objects in the image. Similarly, in NLP, higher order features (n-grams, sub-sequences) can be obtained from lower order features (word embeddings) and the construction of these higher order features is learnt through training the model. A CNN consists of a convolution layer and a pooling layer. The convolution layer uses a filter to generate features and the pooling layer is used to reduce the dimension of the features while preserving their important aspects. The pooling is done to reduce model complexity. We can use multiple filters in one convolution layer to generate different kind of features. Filters are matrices which move along the input data to create feature representations of the data as shown in Figure 2.8. The matrix on the right is the result of matrix multiplication between
the filter and the area in the input matrix that the filter covers.

As shown in the Figure 2.8, a sentence can be represented by a matrix by encoding each word in the sentence by its corresponding embedding (for example, Word2vec or GloVe). Because rows represent words, it is reasonable to use filters with widths equal to the dimensionality of the word vectors rather than a smaller dimension filter. Thus we can simply vary the ‘height’ of the filter or the number of adjacent rows. Each of these filters with different heights can model different length of n-grams.

As shown in Figure 2.8, the matrix representation of the sentence

"Terminals have a gap"

is an input to a convolution layer with multiple filters. The filters have different heights. Each filter produces a different size feature map. For example, a filter of height 4 covers the whole input matrix thus its feature map is of size 1x1. Similarly, a filter of height 2 takes 3 passes to cover the whole of the input matrix, resulting in a feature map of size 3. The number of feature maps generated is equal
to the number of filters used. These feature maps correspond to different n-grams ("terminal-have", "have-a-gap"). The next layer in Figure 2.8 is the pooling layer. Pooling can be done by either taking a maximum value (max-pooling) or taking an average (avg-pooling) of each feature map. The most common form of pooling uses stride 2 together with kernel size 2, which corresponds to partitioning the feature map spatially into a regular grid of square or cubic blocks with side 2 and taking max or average over such blocks for each input feature. In this case, max-pooling is done over each of the feature maps, resulting in 1D feature maps. These single dimension feature maps are concatenated to form a feature vector. The feature vector can then be used further in the network. For example, Zhang and Wallace (2015) use a similar CNN architecture for various sentence classification tasks, such as Sentiment Classification, Question Classification, and Irony Detection. The CNN model outperformed a Support Vector Machine model baseline in each of the datasets. The CNN layers can also be easily stacked to create hierarchical feature maps (Springenberg et al., 2014). Stacked CNN produces a better representative feature for improved model performance.

Sentence comparison tasks like ASAG or text similarity require two CNNs sharing same weights. Each CNN processes a sentence. The feature map of each of the sentences is then combined with specific functions such as vector multiplication, concatenation, or even a feed-forward network. The combined feature map is then used for the final classification. Such a CNN architecture is termed a Siamese CNN Network (Chopra, Hadsell, and LeCun, 2005).

Shao (2017) uses Siamese CNN for the SemEval STS task (Agirre et al., 2012) and was placed first in the competition. A CNN based architecture is also used successfully in the SNLI (Bowman et al., 2015) task. Hu et al. (2014) also use Siamese CNN, Yin et al. (2016) uses a CNN that also uses Attention during composition of CNN feature maps. We will discuss Attention in more detail in next section.
The major issue with CNN for sentence matching task is that the feature extraction of CNN cannot capture long range dependencies (Goldberg, 2016). For example, in the sentence "Terminal 1 and Terminal 2 are separated by a gap", a CNN will only capture a relation between Terminal 1 and gap if the height of the filter is same as the sentence length. Creating large filters increases model complexity, making model computationally very expensive and resulting in slow convergence. It can also lead to over-fitting.

### 2.5.3.2 Recurrent Neural Networks

A Recurrent Neural Network (RNN) (Elman, 1990) is used to model sequences in NLP. For example, a RNN can keep track of the previous words of each of its input words, so a final vector that an RNN models for a sentence will have contextual information between words. At a glance, this is a major improvement over CNN for NLP tasks, where long-range dependencies over words are very important. RNNs preserve these dependencies as the output of a neuron is fed back into itself. An RNN, when unrolled, looks as shown in Figure 2.9.

In Figure 2.9, the left part shows a recurrent neural network and the right part shows its structure when unfolded. From a high level, a RNN is fed an ordered list of input
vectors \( x \in \{x_1, ..., x_T\} \) as well as an initial hidden state \( S_0 \), initialized to all zeros. The RNN returns an ordered list of hidden states \( s \in \{s_1, ..., s_T\} \), as well as an ordered list of output vectors \( y \in \{y_1, ..., y_T\} \). The output vectors may serve as input for other RNN units, when considering deep architectures (multiple RNN layers stacked). The hidden states correspond to the short-term memory of the network. The hidden layer in an RNN is computed as:

\[
s_t = g(Ux_t + Ws_{t-1} + b)
\]

One caveat of a simple RNN is that as the number of words in the sentence increases, the long range effect of other words in the sentence is not captured in the current word. Only a few previous words are tracked to model the next word. This shows that RNN has short-term memory. To overcome this limitation we can use improved RNN architectures such as Long Short Term Memory (LSTM) or Gated Recurrent Units (GRU). Both of these architectures use memory cells and gates to keep track of which information to keep and which to forget. In this way, LSTM and GRU can model long range dependencies with an additional benefit of using only the dependencies that are important for the current task.

As RNN, LSTM, and GRU model contextual information, these architectures are very popular in semantic matching tasks. In a similar way to CNN, RNN architectures are used to encode input sentences. The added benefit to CNN is that now these encoded sentences would also incorporate long range dependencies. For example, in sentence "Terminal 1 and Terminal 2 are separated by a gap", a LSTM/GRU can now capture the relation between Terminal 1 and gap.

Another variation of LSTM is the Bidirectional LSTM (BiLSTM) (Graves, Jaitly, and Mohamed, 2013). A Bidirectional LSTM consists of two LSTMs that are run in parallel: one on the input sequence and the other on the reverse of the input sequence. At each time step, the hidden state of the Bidirectional LSTM is the concatenation
2.5. Deep Learning Methods

2.5.3.3 Attention

Up until now, we have concluded that deep neural networks can be better for the ASAG task than statistical models such as SVM, logistic regression and linear optimisation. This is primarily due to feature extraction and transfer through deep neural networks’ property of deep layers. As is evident from Figure 2.7, we can generate the vector representations of sentences using CNN or RNN architectures and then use these information rich vectors for further processing. In the case of semantic matching tasks, the two vectors from different sentences are combined into a single vector for further processing. If we look at the semantic matching task, and especially ASAG, we can infer that combining the action of vectors can be improved further than just by mere concatenation.

For example, in Figure 2.10, when comparing the student’s answer (bottom) with the reference answer (top), what we are interested in is the alignment between the semantically related terms/phrases such as separated $\rightarrow$ not connected. We can
use attention to encode such relations. Thus, the resulting vector after combining two sentences with attention will also have alignment information in it. The earliest use of attention was in machine translation (Bahdanau, Cho, and Bengio, 2014). This has then adapted to STS tasks (Arroyo-Fernández and Meza-Ruiz, 2017), and SNLI task (Parikh et al., 2016; Kim et al., 2017; Kim, Kang, and Kwak, 2019).

In a high level form, the attention is obtained by first encoding one sentence from the pair of sentences. This encoding can be done using Word2Vec, RNN, or CNN. Now the encoding of second sentence has also access to the encoded first sentence. In this way the second sentence can attend each of the words with only the related word in the first sentence. We will discuss the detailed calculation behind attention in Chapter 3.

The benefit of attention is that the combined vector representation of the two sentences resembles the alignment shown in Figure 2.10.

### 2.6 Conclusion

In this chapter, we have discussed the literature on ASAG. The methods for ASAG can be primarily grouped into methods using templates, alignment, and machine learning. Designing relevant templates and automatically extracting patterns using the templates is an important step forward in ASAG. Automark (Mitchell et al., 2002) and Amati (Willis, 2015) obtained 92% and range of 91% to 98% respectively on their private datasets.

In the case when a reference answer is provided, alignment is an effective method for ASAG. Patterns are extracted from both the reference answer and student response and are matched to grade the student response. Patterns can be a simple words matching (Cutrone, Chang, et al., 2011) or more complex parse tree matching (Siddiqi and
Harrison, 2008). These systems also use semantic information like synonyms to improve results.

The next development in ASAG was to use the various patterns, alignment, and semantic information as features and apply popular machine learning algorithms like SVM or Naive Bayes to grade student responses. The availability of a public ASAG dataset (Dzikovska et al., 2013a) (the SemEval ASAG dataset) paved the way for comparisons between these systems. The systems that showed good results used alignment features that used lexical information and semantic information. Additionally, the systems also used domain adaptation techniques. The domain adaptation technique used the idea that only general features can be ported from one domain to another and domain specific features should be left out. This helped the ASAG system trained in one module also work on different module. Recent work has used a combination of many features used in the semantic similarity task (Sahu and Bhowmick, 2019), and obtained good results in SemEval ASAG dataset (see Table 2.4).

The various machine learning methods showed that using semantic and lexical information and alignments as features are important for the ASAG task. To this end, we explored the deep learning approaches which have used alignment for matching and embedding for using semantic information successfully in other NLP tasks like natural language inference (Parikh et al., 2016) and machine translation (Bahdanau, Cho, and Bengio, 2014).

In the following chapters we will explore how we can adapt the deep learning systems for ASAG and also incorporate ASAG specific information into such models.
Chapter 3

Aligning, Comparing, and Aggregating for Automatic Short Answer Grading

3.1 Introduction

In a setting where reference answers are available, Automatic Short Answer Grading (ASAG) systems grade responses by matching or comparing them with one or more reference answers provided by domain experts or the question setters. The goal of the matching task is to identify the similarity between a reference answer and a student response. A high similarity suggests that the student response is correct and a low similarity suggests that the response is incorrect. For instance, consider Example 3.1, showing a pair of a reference answer and a student response.

Example 3.1

• reference answer Terminal 1 and the positive terminal are separated by the gap.

• student response Terminal 1 and the positive terminal are not connected.
The first sentence is the reference answer and the second sentence is the student response. We can see that the student response is correct as its meaning is similar to the reference answer. Similarly, a student response "Terminal 1 and the positive terminal are connected" is an incorrect response as it contradicts the reference answer and thus is not similar to the reference answer. A student response can also be partially correct, such as for the response, "There is a gap". Such answers have the necessary information to be at least partially correct but not the complete information for it to be wholly correct. Such sentences show a degree of similarity with the reference answer. The ranges of similarity and class of student response is discussed in more detail in later sections.

In this task we are considering two sentences as sequences of words. We then see which words in the reference answer correspond for the purpose of marking task to the words in the student response. We call this process Alignment. The correspondence is not binary but is a measure of degree. We call this measurement similarity. In other words, alignment in this context is the problem of pairing semantically related words or phrases in the student response and the reference answer. Such aligned pairs not only represent relatedness between words or phrases that can be used to aggregate the overall similarity, but they also provides useful information for the explainability of the outcome. Figure 3.1 illustrates the result of an alignment where related terms are paired between the reference answer and student response.
Mitchell et al. (2002) align parse tree representations of reference and student response for grading, while Jordan and Mitchell (2009) use syntactic-semantic templates extracted from reference answer and student response as align candidates. In this chapter, I identify that a non-structural matching followed by a comparison and aggregation step is enough to indicate if a student response is correct or incorrect. I will show that we can achieve good results in the ASAG task with non-structural matching. I will also use this method as a strong baseline for comparison with structural matching. I use a tree representation as align candidates, which I will discuss in this chapter.

First, I identify the use of alignment for a system for grading a student response and then adopt a popular neural network attention mechanism to intuitively tackle the grading problem. The experimental results show that the method outperforms previous benchmarks on multiple datasets. In the context of this dissertation, this chapter explores RQ1 of Chapter 1:

**How can we use the concept of semantic similarity in the ASAG task?**

Additionally, an effective grading system should not only assign an appropriate grade to the student responses but should also be able to explain where that grade comes from, both for auditing grades awarded in assessment (Willis, 2015), and as part of a system for providing feedback (Jordan and Mitchell, 2009). Recently there is high demand for explainable Artificial intelligence (AI) (Gunning, 2017; Samek et al., 2019) for making model’s inference more transparent. In this chapter, I also show how the alignments can be used to explain the grade awarded by the system.

### 3.2 Using Alignment for Grading

As discussed in the introduction, for the ASAG task, each student response is awarded a grade by comparing it with the provided reference answers. To illustrate the ideas,
we will return to Example 1.2 from Section 1.2.

Example 1.2 contains a question, two reference answers, and three student responses. The reference answers are provided by an expert or the question setter. The three student responses show a variety of forms of student response, and marks awarded by human markers. The first response, labelled as ‘C’, is a correct response. It is so marked as its meaning is highly similar to the reference answer 1, in addition to containing all of the information in reference answer 1: Terminal 1 and positive terminal. The second response, labelled as ‘P’, is a partially correct response. It is so marked as it shares a degree of similarity with the reference answer 1. However, it does not contain all the information from the reference answer and thus is only partially correct. The third response, labelled as ‘I’, is an incorrect response. It is so marked as it contains information that contradicts both the reference answers (negative instead of positive terminal) and also does not share high similarity with reference answer 2. Since the third student response is not correct compared to either of the reference answers, it is labelled as incorrect.

The important question now is how do we automatically grade the student response? From our analysis of how the student responses are labelled, we can conclude that we need to represent the reference answer and the student response in a form such that their semantic relationship can be computed. Then we can compute the grade by measuring the semantic relation and also the degree to which the information in the two sentences can be matched (to determine partial correctness). Computing meaning representations and finding a partial match are both challenging tasks. It becomes even more difficult to construct a representation that expresses the entire meaning of complex sentences like the reference sentence in Example 1.2. However, as seen in other semantic matching tasks like entailment, we can align components of two sentences and then compose the overall semantic similarity from component alignments (Parikh et al., 2016). We can use a similar method for ASAG. We can compute the alignment between provided reference answers and a student response.
the result of the computation to predict that the Student #1 response is correct, by simply aligning Terminal 1 with Terminal 1, positive terminal with positive terminal and gap with not connected and recognising that these are synonyms as shown in Figure 3.2. Similarly, we can predict that the Student #3 response is incorrect by aligning positive terminal with negative battery terminal and recognising that these have opposite sense. Finally, we can predict that the Student #2 response is partially correct by aligning gap with gap, recognising that they are the same thing and also have sufficient information for the answer to be at least partially correct. The proposal for partial correctness also implies that the part of the problem is also to decide how much and which alignment is required for the answer to be partially correct.

To solve the problem of alignment, we first extract align candidates. An align candidate is a word/phrase from a sentence that forms a pair with another align candidate from a different sentence. Align candidates are then matched with other align candidates to form aligned pairs. The constraint for aligned pairs is that the participating align candidates exhibit a semantic relatedness (similar, antonym, or meronym) with each other.

Align candidates can be a single word, or a combination of words. In some cases the combination of the words might correspond to syntactic units, for example, a noun phrase: a positive terminal, or an negation relation: not connected. The combination of words can also be non-syntactic, for example, ngrams: terminal
Chapter 3. Aligning, Comparing, and Aggregating for Automatic Short Answer Grading

1 and positive terminal, or sub-sequences: positive-gap. Thus, our proposed steps for a grading system are:

1. Identify align candidates in both the reference answer and student response.

2. Take the align candidates from the reference answer and the student response, and check them for similarity to find aligned pairs.

3. Compare the align candidates in the aligned pairs to compute the relations such as synonymy or contradiction. For example, the system should recognise the relatedness between the word gap and the phrase not connected.

4. For each of the aligned pairs, aggregate all the computed relations between the aligned pairs and then grade the answers accordingly (such as correct, incorrect, or partially correct).

To achieve all the above four steps with high accuracy, we use a neural network that takes a reference answer and student response as input and performs candidate selection, align, compare, and aggregate steps to grade the student response. The neural network uses an end-to-end training mechanism. That is, all four steps are trained simultaneously and only the grade (correct, incorrect, partially correct) of the student response is needed as the label. The primary benefit of this type of training mechanism is that we do not need any alignment specific labels (annotation of align candidates or aligned pairs) on the data, and all three steps help to improve each other during the training.

For model training and evaluation we use the SemEval dataset (Dzikovska et al., 2013a) described in Section 2.3. The detailed statistics of the dataset are provided in Table 2.2 and Table 2.3.
3.3 Structures as Align Candidates

In the previous section we discussed that align candidates can be both single words or tokens carrying structural information. In some cases it is important to model richer structural dependencies in the mode to improve accuracy. For example, we can consider two separate cases: the first case is when there is a high number of alignments between the reference answer and the student response. As shown in Figure 3.3, there is a high number of **aligned pairs** and only a couple of words are not paired. In this case, the student response, “terminal 1 and positive terminal have same state” is an incorrect response but it is marked as a correct response by the word level alignment model as there is a high number of **aligned pairs**.

The second case is of a contradictory **aligned pair** with very few examples in the training data. As shown in the Figure 3.4, the student response has high number of **aligned pairs** with one showing semantic contradiction. The student response is incorrect due to the contradictory alignment. Although this is a notable difference for a human being, for a machine this is only one difference in a sentence pair where the rest of the alignments are correct. Additionally, as there were not enough examples of such contradictions in the training data, the word level alignment model incorrectly marked this student response as correct.

One way to solve such problems is by aligning two sentences by using the structural
Chapter 3. Aligning, Comparing, and Aggregating for Automatic Short Answer Grading

**Figure 3.4:** An example alignment between a reference answer (top) and an incorrect student answer (bottom) with word level alignment. The dashed arrow shows a contradictory pair.

**Figure 3.5:** A structural representation within a sentence. The structural representation is an example and not a standard parse. The structure depicts the relation of gap with other words in the sentence.

representation of those sentences. Figure 3.5 shows an example of such a structural representation of a sentence where the relation of gap is shown with other words in the reference answer. If we then represent the student response with structural relations then the alignment is now computed between structures. The use of such structural units provides a better semantic comparison than using a word-level model (Montemagni and Vanderwende, 1993).

As we can see from Figure 3.6, there are no alignments between the reference answer and the student response. The semantic comparison between the structural elements of the reference answer and the student exhibits zero similarity. With such information, the structural alignment model can correctly predict that the student response is incorrect. The lack of structural alignment points towards the response being incorrect, even though there is a high level of token alignment. A very important point for structural alignment is that we should not extract all the structures from the text. If all the structures are extracted then there will be high alignment between the sentences
3.3. Structures as Align Candidates

Figure 3.6: Structural alignment between a reference answer (top) and a student response (bottom). The alignment is very weak, thus signifying a weaker relation between two sentences.

Shown in Figure 3.6. To avoid extracting all the structures, it is important that the structures are learnt as a part of the alignment process. The learning process can then restrict the number of the structures extracted.

Similarly, if we extract the structural representation of the contradictory case and align it with reference answer as shown in Figure 3.7 we will obtain a very weak relation between the reference answer and the student response. Again, with this information, the structural alignment model can correctly predict that the student response is incorrect.

As mentioned in Section 3.1, word-level alignment can be enough to model inter-sentence relations. However, word based alignment architectures do not directly model any structural relations that may exist among the source elements, and instead rely completely on the hidden layers of the network. While one might argue that structural relations can be learned implicitly by a deep model with enough data, the above discussion shows that in practice it may be useful to provide a structural bias. The structural units can have several forms, such as trees, graphs, or a combination of both. Such structures have been used successfully in various NLP applications. For
example, the use of trees for sentiment analysis and the use of a parser in a neural network model are popular features for text classifications.

Linguistically motivated representations of text structure rely on the availability of annotated corpora as well as a wider range of standard NLP tools such as tokenisers, pos-taggers, and syntactic parsers. Unfortunately, the reliance on labelled data, which is both difficult and highly expensive to produce, presents a major obstacle to the widespread use of the structure for text modelling. Moreover, despite recent advances in structure processing, the use of an external parser often leads to pipeline-style architectures where errors propagate to later processing stages, affecting model performance (Kim et al., 2017).

To this end, there are various techniques to infer structural representations directly from the text without requiring access to a parser or structure-annotated data. In the context of a neural network, the main idea is to first build a hierarchical representation of the text and then use this representation in further layers. In the following sections, we discuss how such a model works and also show its application for ASAG.
3.4 Grading using Deep Neural Networks

The advantage of using a neural network framework over other machine learning frameworks is discussed in detail in Chapter 2, Section 2.5. In summary, for non-neural machine learning techniques like Support Vector Machines (Suykens and Vandewalle, 1999), features need to be identified by a domain expert in order to reduce the complexity of the data and make patterns more visible to learning algorithms. The biggest advantage of deep learning algorithms is that they learn high-level features from data in an incremental manner. This eliminates the need for domain expertise and feature extraction.

Following the steps for grading systems in Section 3.2, we implement a neural network architecture that can model each step of the requirements in its different layers. To the best of our knowledge no such architecture has previously been used in an ASAG task. The input to the neural network is a reference answer and student response in a tokenised form. The output of the network is a label for the student response. In the first step, the neural network identifies the align candidates from the tokenised sentence, then it matches the align candidates to form aligned pairs. The aligned pairs are compared to identify the relations within the aligned pairs. Finally, all the relations are aggregated to grade the student response.

In the following sections, we first discuss the general approach of building a neural network to align, compare, and aggregate for sentence pair classification. We then develop a word based model for the ASAG task. We then extend the word based model to incorporate structure in the input.

For all the above steps we use a typical deep learning methods for text similarity that focuses on creating a single representation for two sentences. A generic example of such a system is shown in Section 2.5.3. In the following section we explain how we represent the different layer shown in Figure 2.7, how we represent the input, how we compute the matching method and how we present the output.
In our experiment, we use the GloVe embedding (Pennington, Socher, and Manning, 2014) to represent words in the sentence. At the time of the experiment, the use of GloVe was due to the better results obtained compared to Word2Vec (Mikolov et al., 2013a). In the first experiment, we do not use LSTM or CNN to encode our sentences. As explained in Section 3.1, word/phrase alignment is enough for ASAG and it does not require complex sentence representations with context information. The matching method according to our requirement includes identifying align candidates and pairing the align candidates. We achieve this by using Neural Attention (Bahdanau, Cho, and Bengio, 2014) (Section 2.5.3.3 for more detail). We then compare the aligned pairs using a feed-forward network (Section 2.5). The aggregation of comparisons is done using concatenation. Then the Softmax layer (Section 2.5.1) gives the final classification. Section 3.4.1 details the architecture of this network.

### 3.4.1 System Description: Align, Compare, and Aggregate (ACA) model

Let \( r = (r_1, \ldots, r_{l_r}) \) represent the reference answer as a sequence of embeddings and \( s = (s_1, \ldots, s_{l_s}) \) represent the student response as a sequence of embedding of lengths \( l_r \) and \( l_s \) respectively.

We assume that each \( r_i, s_j \in \mathbb{R}^d \) is a word embedding vector (see Section 2.5.2) of dimension \( d \).

The training data comes in the form of labelled triplets \( \{(r^n, s^n, y^n) : 1 < n < N\} \) where \( y^n = \{y^n_1, \ldots, y^n_C\} \) is an indicator vector encoding the output label and \( C \) is the number of output classes.

The number of classes is either 2 or 5 depending on the dataset (see Section 2.3). For example, if we are testing 2-way classification, then the output vector for the correct data point is \( \{0, 1\} \) and for incorrect it is \( \{1, 0\} \). At test time, the model receives a pair of sentences \((r, s)\) and the goal is to predict the correct label \( y \).
The core model consists of three components, which are trained jointly: in the first step the elements of $r$ and $s$ are aligned. The alignment is done using neural attention (Bahdanau, Cho, and Bengio, 2014). This step results in a context vector that encodes pairwise alignment between two input sentences, resulting in aligned pairs.

Then each aligned pair is separately compared to produce a set of vectors representing the semantic comparisons of words in the aligned pair. These vectors are aggregated to form one single vector used for classification.

The following sections describes each of the steps in more detail.

### 3.4.1.1 Align Word and Phrases

This step is analogous to measuring the distance between two vectors using the dot product. The alignment of words and phrases is carried out in four steps.

The first step is the *embed* step, where each word in the sentence is represented by a vector as shown in Figure 3.8. As mentioned before, these vectors are embeddings and are extracted through a lookup table. The length of the embedding is a hyper-parameter which can be tuned during model optimisation. A standard length successfully used in many deep learning applications is 300 (Kim, Kang, and Kwak, 2019; Liu et al., 2016).

![Figure 3.8: Embedding a reference answer on the left and student response on the right.](image)

In the second step, these word vectors are encoded into a denser representation. The encoding is done using a feed forward neural network with a fixed output dimension.
Chapter 3. Aligning, Comparing, and Aggregating for Automatic Short Answer Grading

Our experiment uses 200 as the output dimension. The fixed dimension is achieved by using a neural network with output dimension of 200. The weights of the neural network in the encoding steps help tune the embeddings to a particular dataset.

![Encoded vectors](image1)

**Figure 3.9:** Encoding both the reference answer and the student response using a feed forward neural network to tune the vector to the current dataset.

The third step is the attention step. This is the step where we create the aligned pairs. The attention is defined as a random variable $z$ that represents the position in the reference sentence that a word in student response should attend/align to. Then the attention distribution $p$ is the probability of $z$ attaining a position in the reference sentence conditioned on the reference sentence and the current word in student response. This distribution can be parameterised by a feed-forward network $F$.

$$p(z = i| r, s_j) = \text{softmax}([r, s_j])$$

where,

$$[r, s_j] = e_{i,j} = F(r_i)^T F(s_j)$$ (3.1)

where $F(r_i)$ is the encoded transformation obtained from the second step and $F$ is the feed-forward neural network with output dimension of 200 (same as the dimension...
3.4. Grading using Deep Neural Networks

The unnormalised attention matrix is obtained by the matrix wise dot product where $e_{i,j}$ represents the attention weight between a word in the reference answer and a word in the student response. At this stage the since the attention matrix as shown in Figure 3.10 is unnormalised, its row and column does not need to sum to 1.

Finally, in the fourth step the attention weights are normalized to create a probability distribution. Normalizing the attention weights in Equation 3.1, the weighted average of all the token vectors in the reference answer is calculated to obtain a representation of the reference answer that is softly aligned with the token indexed by $i$ in the student response:
Chapter 3. Aligning, Comparing, and Aggregating for Automatic Short Answer Grading

\[ \beta_i = \frac{l_x}{\sum_{j=1}^{l_x} \exp(e_{ij})} \frac{r_j}{\sum_{k=1}^{l_r} \exp(e_{ik})} \]  

(3.2)

Similarly, we compute soft alignment of student response token for each token indexed by \( j \) in the reference answer:

\[ \alpha_j = \frac{l_r}{\sum_{i=1}^{l_r} \exp(e_{ij})} \frac{s_j}{\sum_{k=1}^{l_r} \exp(e_{ik})} \]  

(3.3)

Here \( \beta_i \) is the sub-phrase in \( b \) that is aligned to \( a_i \) and vice versa for \( \alpha_j \). The shape of \( \beta_i \) and \( \alpha_j \) is equal to the embedding size (300 in our case) due to the weighted average computation with the embedding vectors. The sum of the normalised vectors is equal to 1.

If we take GloVe embedding with dimension 100 for words in the reference answer and student response. We get individual beta value for each term in the reference answer with the size of 100. An example computation with GloVe vectors using Equation 3.2 yields:

\[ \beta_{\text{Terminals}} = [0.2591, 0.2430, -0.0220, -1.0451, ..., -0.1000, 0.6393, 0.2259] \]

The value determines the alignment of “Terminals” with the student response. Similarly,

\[ \beta_{\text{gap}} = [-0.1289, -0.0542, 0.6011, -0.4213, ..., -0.3010, 0.8644, 0.2779] \]

The value determines the alignment of “gap” with the student response.

Similarly we also calculate alignment of response terms with the reference answer. Using Equation 3.3, we get:
3.4. Grading using Deep Neural Networks

Figure 3.11: Comparing the aligned pairs

\[ v_{1i} = G([a_j, b_i]) \]
\[ v_{2j} = G([b_j, a_j]) \]

\[ \beta_{\text{both terminals}} \]
\[ \beta_{\text{both have}} \]
\[ \alpha_{\text{gap}}, \text{connected} \]

Normalise

\[ \beta_i = \frac{\exp(e_{ij})}{\sum_{k=1}^{l} \exp(e_{ik})} \]
\[ \alpha_i = \frac{\exp(e_{ij})}{\sum_{k=1}^{l} \exp(e_{ik})} \]

<table>
<thead>
<tr>
<th>Both</th>
<th>0.03</th>
<th>0.06</th>
<th>0.1</th>
<th>0.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.02</td>
<td>0.2</td>
<td>0.2</td>
<td>0.04</td>
</tr>
<tr>
<td>terminals</td>
<td>0.9</td>
<td>0.01</td>
<td>0.04</td>
<td>0.2</td>
</tr>
<tr>
<td>are</td>
<td>0.001</td>
<td>0.3</td>
<td>0.1</td>
<td>0.04</td>
</tr>
<tr>
<td>connected</td>
<td>0.049</td>
<td>0.02</td>
<td>0.03</td>
<td>0.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Terminals</th>
<th>have</th>
<th>a gap</th>
</tr>
</thead>
</table>

\[ \alpha_{\text{Both}} = [1.5e - 01, 6.50e - 01, 1.9e - 03, \ldots, -5.0e - 01, 6.4e - 01, 2.5e - 01] \]

The value determines the alignment of “Terminals” with the student response. Similarly,

\[ \alpha_{\text{connected}} = [7.1e - 02, 5.2e - 01, -2.5e - 03, \ldots, -5.7e - 01, 6.9e - 01, 3.7e - 01] \]
Chapter 3. Aligning, Comparing, and Aggregating for Automatic Short Answer Grading

3.4.1.2 Compare the Aligned Elements

This step computes the semantic relation between the aligned phrases. The semantic relation could be synonymy, antonymy, among others.

Each aligned phrase \( \{(r_i, \beta_i)\}_{i=1}^{l_r} \) and \( \{(s_j, \alpha_j)\}_{j=1}^{l_s} \) is separately compared with each other using a function \( G \). This function is again a feed-forward network:

\[
\begin{align*}
v_{1,i} &= G([a_i, \beta_i]) \\
v_{2,j} &= G([b_j, \alpha_j])
\end{align*}
\] (3.4)

where the brackets \([,]\) denote concatenation. \( v_{1,i} \) is the comparison between token \( i \) in the reference answer and all the student response tokens that are softly aligned with token \( i \). Similarly, \( v_{2,j} \) is the comparison between token \( j \) in the student response and all the reference answer tokens that are softly aligned with token \( j \).

3.4.1.3 Aggregate the Comparisons

The comparison step results in two sets of comparison vectors \( \{v_{1,i}\}_{i=1}^{l_r} \) and \( \{v_{2,j}\}_{j=1}^{l_s} \).

Now we combine the effect of each individual comparison for the final classification. The step is termed as aggregation. Each of the comparison sets is aggregated by summation.

\[
\begin{align*}
v_1 &= \sum_{i=l_r}^{l_r} v_{1,i} \\
v_2 &= \sum_{j=l_s}^{l_s} v_{2,j}
\end{align*}
\] (3.5)

The concatenation of the aggregated vectors is fed through a final classifier \( H \), that is a feed forward network followed by a linear layer:

\[
\hat{y} = H([v_1, v_2])
\] (3.6)
3.4. Grading using Deep Neural Networks

Aggregation

\[
\begin{align*}
 v_1 &= \sum_{i=1}^{L} v_{1,i} \\
 v_2 &= \sum_{j=1}^{S} v_{2,j} \\
 v_{3,i} &= G([a_j, \beta_j]) \\
 v_{2,j} &= G([b_j, \alpha_j])
\end{align*}
\]

Figure 3.12: Aggregating the compared pairs.

where \( \hat{y} \in \mathbb{R}^C \) represents the predicted (unnormalized) scores for each class and consequently the predicted class is given by \( \hat{y} = \arg \max_i \hat{y}_i \).

The training uses multi-class cross-entropy loss with dropout regularization (Srivastava et al., 2014):

\[
L(\theta_F, \theta_G, \theta_H) = \frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} y_c^{(n)} \log \frac{\exp(\hat{y}_c)}{\sum_{c'=1}^{C} \exp(\hat{y}_{c'})}
\]

(3.7)

where, \( L\theta_F, \theta_G, \) and \( \theta_H \) are learnable parameters of the functions \( F, G, \) and \( H \) respectively. \( N \) is the number of training data and \( C \) is the number of class.

All of the modules are trained in an end-to-end learning architecture. End-to-end (E2E) learning refers to training a possibly complex learning system represented by a single model (specifically a Deep Neural Network) that represents the complete target system, bypassing the intermediate layers usually present in traditional pipeline designs (Collobert et al., 2011). A clear limitation of this pipelined architecture is that each module has to be optimized separately under different criteria. The E2E approach consists of replacing the pipelined chain for a single Neural Network, allowing the use of a single optimization criterion for enhancing the system. An added advantage of the E2E architecture is that the error is propagated from the final layer of the network to the starting point of the network and is corrected at each layer.
Chapter 3. Aligning, Comparing, and Aggregating for Automatic Short Answer Grading

Figure 3.13: The architecture of attend, compare, and aggregate model from bottom to top.
3.4. Grading using Deep Neural Networks

(Rumelhart et al., 1995). In our case this means that during training, the output loss at the aggregate layer forces a correction at the comparison layer, which in turn forces a correction at encoding layer. The overall architecture is shown in Figure 3.13.

3.4.2 System Description: Structural Align, Compare, and Aggregate (SACA) model

In this section, we will discuss an approach to obtain a structural representation of sentences as shown in Figure 3.5. The approach follows the work by Kim et al. (2017). Kim et al. used the structural representation of input sentences to improve results on a bag of words representation for the machine translation and language inference tasks.

The difference between the ACA and SACA models is in the representation of the input to the encode layer. The SACA model builds a structural representation and provides that as an input to the embed layer instead of just using the GloVe representation as the input.

To model structural relations between words we can use the same attention mechanism as shown in the ACA model but instead of using attention between two sentences we use attention within the sentence (intra-sentence attention). This, in turn, gives a relation between words within the sentence as shown in Figure 3.14.

We will first look at how to use the inter-sentence attention method to develop a word-to-word relation within a sentence and then expand it further to develop a tree-type relation within a sentence.

Word-to-word Relation for Sentences

We can use the attention mechanism for representing individual sentences. The key idea is to capture the interaction between tokens within a sentence, generating a context representation for each word with structural information.
Calculating attention weights is similar to that shown in Chapter 3. The only difference is that instead of calculating inter-sentence attention we calculate intra-sentence attention. In other words, we are calculating attention between words of the same sentence instead of attention between words of two sentences.

Given a sentence $s$ represented as a sequence of $n$ word vectors $[r_1, r_2, ..., r_n]$, for each word pair $(r_i, r_j)$, the attention score $e_{ij}$ is estimated as:

$$f_{ij} = F(r_i, r_j)$$

$$e_{ij} = \frac{e(f_{ij})}{\sum_{k=1}^{n} e(f_{ik})}$$

where $F$ is a function learnt from a feed-forward network for computing the unnormalised score $f_{ij}$, a relation between word $i$ and word $j$, which is then normalised by calculating a probability distribution $e_{ij}$. Individual words collect information from their context based on $e_{ij}$ and obtain a context representation:

$$u_i = \sum_{j=1}^{n} e_{ij} r_j$$
where the attention score $e_{ij}$ indicates the (dependency) relation between the $i^{th}$ and the $j^{th}$ words and how information from $r_j$ should be fed into $r_i$. Following from Chapter 3, the normalised attention distribution can be viewed as a distribution over a variable $z \in [1, ..., n]$, that represents the word position to be attended to. For example, $p(z = 2 | s, r_3)$ gives the probability value of relation between the word $u_3$ and $u_2$.

In the context of ASAG, such intra-attention is obtained for both the student response and the reference answer. The intra-sentence representation of both the texts is then passed further into the inter-sentence attention model. In doing so the inter-sentence attention now represents the structural alignment as depicted in Figure 3.5.

Although at this point the intra-sentence attention does induce some structural forms, these forms are shallow and limited to word to word dependencies. Since attention is computed as a simple probability distribution, it cannot capture more elaborate structure dependencies such as trees or graphs. Since $z$ is a single variable distribution, each word can have a strong relation to only one other word in the sentence, as shown in Figure 3.15. For example, if we take the sentence "Terminal 1 and positive terminal have a gap", the word gap with simple attention can have a strong connection with only one other word (say Terminal). In this sense we miss the relation between gap and positive terminal.
Figure 3.16: Random variable representing distribution to compute a structured attention network. ’S’ is the source word for attention.

Figure 3.17: Phrase structure parse of a sentence (obtained from AllenNLP (Joshi, Peters, and Hopkins, 2018)).

Figure 3.18: Dependency parse of a sentence (obtained from AllenNLP (Dozat and Manning, 2016)).

Tree Structure for Sentences

To overcome the limitation of word-to-word relations, we capture structures like trees or graphs. For this we define the attention distribution \( (z) \) in a form that can model trees or graphs. The popular notions of trees or graphs widely used in NLP are dependency trees and phrase structure trees (Hudson and Hudson, 2007). In dependency trees, all nodes are labelled with words. In phrase structure trees, only leaf nodes are labelled with words, with the internal nodes’ labels representing phrasal categories. For the ASAG task, we want to extract structures that are based on the
3.4. Grading using Deep Neural Networks

Figure 3.19: Tree structure showing relation between words.

direct relation between words like dependency trees.

For example, as we can see in Figure 3.18, gap has a relation with terminal 1 and negative terminal through the word have. Thus, when we align have with words in another sentence, it carries its relations with both gap and terminal 1 and negative terminal. However, if we use phrase trees then the sentence partitions into sub-phrases and thus the alignment would only between the sub-phrases without any relation between them (as seen in Figure 3.17). Thus, we use the concepts relating to the dependency trees to extract structures from reference answers and student responses. The structures are learnt from the data without any prior annotation. Since we are not using any prior annotations, the extracted structures may not have the same structures as a dependency tree. Not having the same structures as the dependency tree is a desired behaviour. We need this behaviour as we want to extract structures that improve the accuracy of the model we are building.

We follow the rule for graphical models to generate dependency tree-like structures (Smith and Eisner, 2008). The rule has two-parts: the first part defines a parent or head word and the second part makes sure that each structure has only one head. The two rules for the graphical model are represented as:

1. latent variables $z_{ij} \in \{0, 1\}$ for all $i \neq j$, which indicates that the $i^{th}$ word is the parent of the $j^{th}$ word ($x_i \rightarrow x_j$)

2. a special global constraint that rules out configurations of $z_{ij}$ that would violate parsing constraints (one head).
Thus, instead of defining $z$ as a single random variable as in word-level attention, the attention distribution is a collection $[z_{11}, ..., z_{nn}]$, as shown in the Figure 3.16. Using the collection of a random variable, now each word in a sentence can have a strong relation with more than one word. For example, `gap` can have a strong relation with `Terminal` and `Positive terminal`, thus representing a tree architecture as shown in Figure 3.19. The probabilities for each sub-tree of the parse cannot be calculated by a simple dot product. For this, we follow the implementation of intra-sentence structural alignment presented by Kim et al. (2017). The authors calculate the probability distribution for the collection of $z$ using the inside-outside algorithm (Baker, 1979).

**Inside-outside Algorithm**  The inside-outside algorithm (Baker, 1979) gives the probability of the substructures at each position in the sentence. The inside-outside algorithm is used to generate probabilities for a probabilistic grammar. The input to the inside-outside algorithm is sentences, grammar rules, and the weights associated with the rules. The weights associated with rules determine the most likely parse to generate for each sentence. To start with, these weights are randomly assigned. The Inside-Outside algorithm starts with initial random weights and iteratively adjusts them so that the likelihood of the training corpus (in this case the given sentences) increases. The Inside-Outside algorithm is a special case of the Expectation Maximisation (EM) algorithm (Dempster, Laird, and Rubin, 1977) for maximum likelihood estimation of models. However, it is beyond the scope of this dissertation to describe in detail how the Inside-Outside algorithm derives from the EM algorithm.

In the context of the neural network architecture that we use to generate structures, the grammar rules are followed from Smith and Eisner (2008) (the two rules mentioned above) and the initial weights are calculated using the simple intra-sentence alignment as mentioned in word-to-word attention (using a single random variable $z$). The algorithm then outputs the probability values for each subtree within the
sentence’s parse tree. The most important part of the inside-outside algorithm is that
the algorithm is differentiable (Eisner, 2016) and thus can be trained using back-
propagation in the neural network.

**Intra-sentence alignment: Structured Alignment**

Structure alignment is computed within the words in a reference answer and in a
student response, using intra-sentence attention as described above.

First, the unnormalised attention score is calculated as:

\[
 f_{ij} = F(r_i, r_j)
\]

where \( F() \) is a combination of two Long Short-Term Memory (LSTMs) (Hochreiter
and Schmidhuber, 1997) and a dot product. The first LSTM reads the sentence in a
forward direction and the second LSTM reads the sentence in the backward direction.
Such a structure is commonly known as the BiLSTM (Chen et al., 2017). A BiLSTM
incorporates a forward LSTM layer and a backward LSTM layer to learn information
from the preceding as well as the following tokens of a sentence. These scores are
used as input to the inside-outside algorithm to obtain the probability of each word’s
parent. The probabilities \( w \) is then used to obtain the parent for each word. Thus the
unnormalised \( f_{ij} \) is normalised into a probability distribution as:

\[
 e = \text{insideOutside}(f)
\]

where the inside-outside algorithm constrains the probability distribution \( e_{ij} \) to be the
posterior marginal of a dependency tree structure.
Finally, we use the normalised structured attention, to build a context vector for updating the semantic vector (encoded vector) of each word, obtaining new representations as:

$$str_i = \sum_{j=1}^{n} e_{ij}r_j$$

The above process of obtaining a structured representation of a sentence is shown in Figure 3.20.

The same process is also carried out for student responses. This obtains their context vectors, including their structure attention representation. The model then follows the same architecture as the ACA model from the Encode step.

### 3.5 Experiments

The dataset is tokenised using the Stanford parser (Manning et al., 2014). All the modules are implemented using TensorFlow (Abadi et al., 2016). We used unseen answers as the development set and unseen questions as a test set and vice-versa. For unseen domains, we used both unseen answers and unseen questions as the development set. The development set in each instance is used for selecting the best model. Table 2.2 shows the number of pairs for training, unseen answers, and unseen questions. The number of pairs is the same for both 2-way and 5-way classification.

The development set is used for hyper-parameter exploration. The best result for both ACA and SACA model in the development set is found with 300 hidden units, 0.7 dropout keep probability, 0 l2 loss, a batch size of 32, a learning rate of 0.05 and the Adagrad optimiser algorithm (Duchi, Hazan, and Singer, 2011). The words are embedded using a 300-dimensional GloVe embedding (Pennington, Socher, and
Figure 3.20: Generation structure alignment for a sentence using a BiLSTM and the inside-outside algorithm.
Chapter 3. Aligning, Comparing, and Aggregating for Automatic Short Answer Grading

The weights of the neural network were initialised randomly. Performance of the models were evaluated using weighted average F1 scores (see Section 2.3.1 for more detail). The F1 scores presented are the average of 10 runs of the experiment. The random initialisation and the average of 10 runs of the experiment is same for rest of the experiments in this thesis.

### 3.5.1 Results for the Development Set

Both unseen answers and unseen questions are separately used in the development set. Table 3.1 shows the best weighted F1 score obtained during the training of the development set. The alignment, compare, and aggregate model is shown as ACA in the Table 3.1.

The results are compared with that of CoMeT (Ott et al., 2013) as it is a non-neural system that also uses alignment, comparison, and aggregation.

The ACA model outperforms the CoMeT in every category, except on unseen answers, where CoMeT has a better score by 3 points weighted F1. The high scores for unseen answers compared to the unseen question is expected: the unseen answers have higher similarity to the training pairs as they are based on the same questions. Similarly, because the unseen questions generates completely different answers, the accuracy of the system goes down.

As mentioned earlier, the ACA model did not use any handcrafted features as in CoMeT. The results obtained are a strong motivating factor for using deep learning in

<table>
<thead>
<tr>
<th>Dataset type</th>
<th>ACA dev-uA</th>
<th>ACA dev-uQ</th>
<th>CoMeT (Ott et al., 2013) dev-uA</th>
<th>CoMeT (Ott et al., 2013) dev-uQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-way</td>
<td>0.85</td>
<td>0.78</td>
<td>0.88</td>
<td>0.75</td>
</tr>
<tr>
<td>5-way</td>
<td>0.72</td>
<td>0.73</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

TABLE 3.1: Weighted F1 score obtained with the best model in development set and comparison with development set weighted F1 obtained from CoMet and Riordan et al. (2017).
this context, as the features for alignment and classification are learnt automatically.

3.5.2 Results for the Test Set

In Table 3.2 and Table 3.3, we compare our ACA and SACA models with the best performing models in the SemEval dataset. The models are ETS run-1, 2, and 3 (Heilman and Madnani, 2013), CoMeT (Ott et al., 2013), SoftCardinality (Jimenez, Becerra, and Gelbukh, 2013), and Feature Engineering (Sahu and Bhowmick, 2019). The details of the models are given in Section 2.3.1. For Feature Engineering, authors only provide results for 5-way classification.

Table 3.2: Weighted F1 for the 2-way classification for the ASAG models.

<table>
<thead>
<tr>
<th>System</th>
<th>Beetle UA</th>
<th>Beetle UQ</th>
<th>SciEntBank UA</th>
<th>SciEntBank UQ</th>
<th>SciEntBank UD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoMeT</td>
<td>0.839</td>
<td>0.702</td>
<td>0.773</td>
<td>0.577</td>
<td>0.677</td>
</tr>
<tr>
<td>ETS1</td>
<td>0.810</td>
<td>0.732</td>
<td>0.714</td>
<td>0.703</td>
<td>0.694</td>
</tr>
<tr>
<td>ETS2</td>
<td>0.840</td>
<td>0.715</td>
<td>0.770</td>
<td>0.622</td>
<td>0.574</td>
</tr>
<tr>
<td>ETS3</td>
<td>0.845</td>
<td>0.711</td>
<td>0.767</td>
<td>0.580</td>
<td>0.528</td>
</tr>
<tr>
<td>SoftCardinality</td>
<td>0.782</td>
<td>0.652</td>
<td>0.722</td>
<td>0.745</td>
<td>0.712</td>
</tr>
<tr>
<td>ACA</td>
<td>0.826</td>
<td>0.802</td>
<td>0.783</td>
<td>0.745</td>
<td>0.701</td>
</tr>
<tr>
<td>SACA</td>
<td>0.831</td>
<td>0.810</td>
<td>0.791</td>
<td>0.758</td>
<td>0.725</td>
</tr>
</tbody>
</table>

Table 3.3: Weighted F1 for the 5-way classification for the ASAG models.

<table>
<thead>
<tr>
<th>System</th>
<th>Beetle UA</th>
<th>Beetle UQ</th>
<th>SciEntBank UA</th>
<th>SciEntBank UQ</th>
<th>SciEntBank UD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoMeT</td>
<td>0.675</td>
<td>0.445</td>
<td>0.598</td>
<td>0.299</td>
<td>0.252</td>
</tr>
<tr>
<td>ETS1</td>
<td>0.552</td>
<td>0.547</td>
<td>0.535</td>
<td>0.487</td>
<td>0.447</td>
</tr>
<tr>
<td>ETS2</td>
<td>0.705</td>
<td>0.614</td>
<td>0.625</td>
<td>0.356</td>
<td>0.434</td>
</tr>
<tr>
<td>ETS3</td>
<td>0.700</td>
<td>0.586</td>
<td>0.640</td>
<td>0.411</td>
<td>0.414</td>
</tr>
<tr>
<td>SoftCardinality</td>
<td>0.558</td>
<td>0.450</td>
<td>0.537</td>
<td>0.492</td>
<td>0.471</td>
</tr>
<tr>
<td>Feature Engineering</td>
<td>0.709</td>
<td>0.6248</td>
<td><strong>0.925</strong></td>
<td>0.658</td>
<td>0.656</td>
</tr>
<tr>
<td>ACA</td>
<td>0.728</td>
<td>0.671</td>
<td>0.662</td>
<td>0.653</td>
<td>0.630</td>
</tr>
<tr>
<td>SACA</td>
<td>0.731</td>
<td><strong>0.674</strong></td>
<td>0.686</td>
<td><strong>0.663</strong></td>
<td><strong>0.661</strong></td>
</tr>
</tbody>
</table>

From Table 3.2 and Table 3.3, we can see that the SACA model outperforms all the other models except for the unseen answers type for Beetle in the two-way classification and unseen answers type for SciEntBanks in 5-way classification. The structured
Chapter 3. Aligning, Comparing, and Aggregating for Automatic Short Answer Grading

<table>
<thead>
<tr>
<th></th>
<th>Beetle</th>
<th>SciEntsBank</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UA</td>
<td>UQ</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>2-Way</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACA</td>
<td>0.83</td>
<td>0.82</td>
</tr>
<tr>
<td>SACA</td>
<td>0.84</td>
<td>0.83</td>
</tr>
<tr>
<td>5-way</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACA</td>
<td>0.77</td>
<td>0.72</td>
</tr>
<tr>
<td>SACA</td>
<td>0.77</td>
<td>0.73</td>
</tr>
</tbody>
</table>

**Table 3.4:** Precision and Recall values for ACA and SACA models for 2-way and 5-way classification.

The attention model also outperforms the Feature Engineering model in all categories except the unseen answers category for the SciEntsBank dataset. We believe that the very high score for the unseen answers category for the Feature Engineering model is due to its relevance-feedback feature. The relevance-feedback feature helps when the questions are same, since adding back student responses to model answers covers a large concept space. However, this does not transfer equally when either the questions or the domains are unseen, since in these cases, adding student responses to the model answers does not cover newer concepts seen in unseen questions or unseen domains.

It should be noted that the ACA model outperforms the previous best models in the task, and the SACA model improves on the ACA model. The improvement is significant in the case of unseen questions in Beetle and unseen questions and unseen domains in SciEntsBank. Also, these models use a very narrow range of language features compared to other state-of-the-art models.

The SACA model learns generic features from the grading dataset that apply to new questions as well as previously seen questions.

The higher scores for unseen questions compared to other models are encouraging for adapting neural models with embeddings. We suspect that the low score of non-neural model in unseen questions could be due to the failure of similarity functions
3.5. Experiments

Figure 3.21: Confusion matrix for ACA and SACA model. The labels 0, 1, 2, 3 and 4 represent correct, partially-correct, contradictory, irrelevant, and non-domain category respectively.

Figure 3.21 shows the confusion matrix obtained from the best performing model for both ACA and SACA. From the confusion matrix, we can see that SACA model performs generally better than ACA model. The performance improvement is notable in the unseen answer and non-domain category. The matrix also shows high misclassification between correct and partially correct categories for both models.

Table 3.4 shows precision and recall for the 2-way and 5-way classification. We can see that the model mostly shows a balanced precision and recall values while slightly
favouring precision over recall in a few categories.

3.6 Discussion

One of the motivating reasons for using the ACA model was to be able to interpret the results to assist human graders. The alignment aspect of the model is particularly pertinent here, to get a general idea of why a given student response has been classified as correct or incorrect. The alignments are investigated through heat-maps as shown in Figure 3.22. The heap-maps depict the normalised attention weights of the ACA model that are the resulting alignments. The alignment score ranges from 0 to 1, where 1 is the maximum alignment. Sentence 1, the model answer, is shown on the y-axis and words from it are subscripted with 1. Sentence 2, the student response, is shown on the x-axis and words from it are subscripted with 2.

The role of embeddings for correct prediction The left heat-map in Figure 3.22 shows the alignment map for the model answer with a correct student response. The model correctly predicted the result. We can see that it correctly identifies the alignment of $gap_2$ and $separated_2$. This is mainly due to the embeddings that provide
enough information for the relation of \(gap_1\) with \(separated_2\) and the comparison module in the network that processes the alignment for the correct prediction.

Similarly, as shown in the right heat-map of Figure 3.22, for the incorrect answer, the model shows an alignment of \(gap_1\) and \(connected_2\). Although the compare value of the model cannot be discerned, it can be assumed that the model aligns these words as contradictory pairs.

The network not only does the comparison between two words but it also does the comparison with the combination of words. The benefit of this can be seen in Figure 3.23, where two words, \(closed_1\) and \(path_1\), are aligned with the word \(connected_2\) for comparison. This alignment allows the network to correctly predict the grade for the student’s answer (correct in this case).

**Shortcomings**  One of the obvious shortcomings of the neural network approach is that it depends on the number of examples used. If the number of data examples supplied is inadequate, then the network will fail to learn distinguishing features. For example, in Figure 3.24, the alignment fails to capture the negation in the \(connected_1\) part and since the model has presumably learnt that \(gap\) and \(connected\) are antonyms,
Figure 3.24: Alignment map where negation is not detected.

Figure 3.25: Alignment that ignores important detail ($change_1$) from the reference answer
it produces an incorrect result. This does not imply that increasing the size of the dataset is guaranteed to produce better performance in the network. It merely highlights the problem that neural networks do not work well with a limited amount of data. This is also seen with high similarity between “each” and “positive”. The similarity value is determined by Equation 3.1. The GloVe vector is first transformed by an embed function before calculating the cosine similarity. Without the embed function, the GloVe similarity between “each” and “positive” is 0.42 (green zone in the heat-map). In our context if we had enough examples then the embed function would reduce this similarity to a lower value.

Additionally, we suspect that the performance of the neural network may be affected by the lack of consideration of word order in the model. A model which was more aware of word order might be able to correctly interpret the effect of negation such as not in front of another word.

For the reference and student response pair:

**reference answer:** When there is a change in voltage you have found the damaged bulb.

**student response:** There will be a voltage reading where the bulb is burned out.

the model fails to identify the important details in the sentence. This is also shown in the alignment in Figure 3.25. Here, the word change is aligned to voltage reading but the word change is essential for the answer to be correct. The model fails to identify the essential element and labels the answer as correct. This can be due to the lack of word order information in the model, and also due to the inability of the model to learn to distinguish important and unimportant features from the
sentences. Without these, the answer becomes incomplete and incorrect. Identifying important words is critical to automatic grading of students. Such words can be the difference between the correct and incorrect student response, and so we are tagging this as a key focus for another chapter.

Similar conclusions can also be seen in the variety of examples from Section 2.3. Both the ACA model and SACA model correctly identified the correct, partially correct, and incorrect unseen responses from Example 2.3. We should note that the incorrect response has contradictory information, and the alignment models accurately detected the contradiction. Similarly, the models also correctly detected the contradiction in incorrect response of Example 2.7.

For the unseen question from the Beetle dataset in Example 2.4, the SACA model correctly classified all the three responses but, the ACA model classified the partially correct response as a correct response. This is due to the lack of structural relationships between the terms “gap” and “circuit” in the ACA model. The structural matching of the SACA model correctly identifies the partial correctness of the response.

Both the ACA the SACA models incorrectly identified the partially correct response in Example 2.6 as an incorrect response. Example 2.6 is an unseen domain question thus the models did not have enough information about the response and the reference answer during the training process.

Similar output is also seen in the unseen domain category in Example 2.7 where both the model correctly identified the correct and the incorrect response but failed to identify the partially correct response. The correct response has enough matching elements with the provided reference answer and thus is identified correctly by the models. However, the partially correct response does not have sufficient information, and since similar examples are never seen during training, both the models incorrectly classified the response as an incorrect response.
3.7 Conclusion

The chapter presents the benefits of an efficient state-of-the-art neural network applied to the task of automatic short answer grading, and given the results of applying the system to a well-known dataset in that domain. The approach outperforms a state-of-the-art technique that uses manually crafted features in a machine learning setting. The added advantage of the ACA model is that the alignments can be extracted from the model to assist human graders to understand the network’s behaviour for grade allocation. We used heat-maps to show the working of the network and the alignment between words and phrases. The ACA model showed very competent performance and along with the knowledge of embeddings and its alignment and comparison framework, it managed to show a relationship between a reference answer and student response. This relationship is very useful for understanding the final prediction. The SACA model captures relationship between words and creates appropriate structural units within a sentence. Matching of such structural units between sentences provides improved similarity calculation that improves performance in the ASAG task.

Although the models produced good results for 2-way classification, its 5-way classification still needs improvement compared to a similar entailment task, in which the result of multi-way classification is significantly higher (Parikh et al., 2016), although this could be the result of the more complex grading dataset. Identifying the most important words that characterise the correct comparisons between the two sentences could be important to improve model performance. Finding the most important word is identified as important in Willis (2015) for ASAG. Additionally, it will also be interesting in modelling negation directly in the neural network setting (Socher et al., 2013) to further improve the results.
Chapter 4

Incorporating Key Information for ASAG

4.1 Introduction

In Chapter 3 we used an attention network to model word level and structural relations between a reference answer and a student response. We also showed that this is an effective approach for inferring the semantic relationship between two sentences. However, analysing our results indicates that we could improve the performance of the system if we were able to distinguish between important and unimportant features in the reference answers and students’ responses. For example, let us consider the reference answer and (incorrect) student response pair taken from the unseen question category:

**Example 4.1**

*Reference Answer:* When there is a change in voltage you have found the damaged bulb.

*Student Response (I):* There will be a voltage reading where the bulb is burned out.
Both the word level and the structural model classified the response in Example 4.1 as the correct answer. The word level model aligns the word *change* with the phrase *voltage reading* and the structural model aligns the phrases *change voltage* and *voltage reading*. Both models conclude that these alignments are enough for the response to be correct. Since the response is incorrect, both the models are wrong in this instance. Both models are incorrect because they have failed to recognise the most important details in the sentence. The correct response in this context is to specify that there is a *difference* in voltage reading as signified by the word *change* in the reference answer. Thus, the word *change* (or similar words) is essential for the answer to be correct. Identifying important phrases is critical to automatic grading of students. Such words can be the difference between the correct and incorrect student response. We term such important phrases Key Information (KI). KI is new information that is not present in the question.

We also saw the importance of KI in Example 7, where the correct and partially correct responses contained KI like *gap* and *connected* and the incorrect response did not contain KI.

Empirically (see Section 4.5) we show that KI is very useful for correctly classifying student responses. An important aspect of key information is that without the key information the answer must be incorrect. For example, "*there is a presence of terminal 1 and terminal 2*" is an incorrect response as it does not contain the key information *gap* or terms similar to *gap*. Key information is necessary for the answer to be correct but it is not a sufficient condition as we can see from Student #3’s response in Example 1.2. The response does contain the key information *gap* but is still incorrect as it contains *negative battery* instead of *positive terminal*. Thus in addition to the key information we also need an
alignment mechanism to check the semantic equivalence of the student and reference answers. In Figure 4.1, the key information is shown as bold text.

The semantic alignment between the student response and the reference answer gives us an estimate of how similar in meaning the student response is to the reference answer. The key information can then supply vital knowledge for the cases:

1. where there are substantial alignments with a reference answer but the correct relation with key information is lacking (as in Student #3’s incorrect response) in Example 1.2, and

2. where the number of alignments is low but the answer does contain the key information (as in Student #2’s partially correct response) in Example 1.2.

In addition to these cases, key information can also provide an additional level of confidence for marking correct answers (Student #1’s answer) in Example 1.2 or identifying incorrect answers that do not contain key information.

Previous work on comparing student responses against reference answers has used techniques like minimal meaningful propositions (Godea, Bulgarov, and Nielsen,
Chapter 4. Incorporating Key Information for ASAG

2016) and focus (Ziai and Meurers, 2014). Both approaches extract chunks that provide a form of a complete answer to the question. These chunks are manually annotated in the data. Key information is a word or chunk (contiguous or non-contiguous) in an answer that plays a vital role in deciding the correctness of an answer. Key information does not need to be meaningful propositions. Key information is less constrained than meaningful propositions in the sense that although it may be required for the answer to be correct, it in itself may not be a correct answer.

For example, if gap were represented by not connected in Example 1, both minimal meaningful propositions and focus are represented as not connected but in our case, the key information is just connected. Such key information can then be combined with a structural alignment model to infer knowledge on not. This nature of key information enables us to extract key information with a simple and lightweight approach (see Section 4.3) which does not require manual annotations but requires more than one reference answer to generate key information.

This chapter introduces the notion of Key Information (KI) for semantic text similarity in the context of Automatic Short Answer Grading. The chapter also gives the details of an algorithm for identifying key information. The extracted key information is used in a neural structured alignment model which uses the key information to improve the accuracy of grading.

4.2 The importance of Key Information for Automatic Grading

For the automatic marking dataset, the alignment models discussed in Chapter 3 work on the principle that if there is high alignment between a reference answer and the student’s response, then the student’s response should be marked as correct. Similarly, if there is low alignment then the response should be marked as incorrect
or partially correct. This means that although the model does have the ability to learn a perfect or a partial match, it does not indicate whether the partial match is due to key information or non-key information. Generally, in a machine learning context, the difference between the key and non-key information can be learned from the data, but if the size of training data is not sufficient, then the model can resort to the number of matches between the reference answer and the student response. This is especially true for scenarios where either the questions or even the domain are not represented in the training data. In these cases, the model may not have enough information to mark new cases. For example, an alignment system could mark a student’s (correct) response “because there is a gap” as incorrect, as there are not enough matches with the reference answer in the training data.

To overcome the limitation of an alignment model’s inability to separate key information from non-key information, we can provide that additional key information to the model. Providing key information can help a model to grade student responses, depending on whether the key information is present or absent. In the case when there is a strong alignment with the reference answer but the key information is missing, the model then can mark the response as incorrect.

Key information becomes especially important to mark responses accurately for questions in new domains. Although the model is not trained on similar answer/response pairs, the grammatical structures learned by the alignment model can be carried over to responses in the new domain, while the KI provides the domain-specific knowledge. In other words, key information can help with domain adaptation. One of the previous successful systems in the Beetle dataset used domain adaptation to achieve better results in the unseen question categories (Heilman and Madnani, 2013). The system however does not use the concept of key information but uses text similarity as a feature for domain adaptation. In Section 4.5.2, we show that KI can help with domain adaptation.
Additionally, key information helps to correctly classify student responses when the alignment models only show a small number of alignments between a reference answer and the student response. For example, even though there may be few alignments with the reference answer, the presence of the key information in the student response can indicate to the model that the response is still correct.

In the following sections, we describe our approach that utilises key information for ASAG. First we extract key information from the data and then use the extracted key information to build upon our model from Chapter 3.

### 4.3 Key Information Extraction (Extractor)

The intuition in key information extraction is that key information is new information that the student’s answer needs to provide in order to be marked as correct. Such new information is not present in the question. When multiple reference answers are provided, typically the key information occurs repeatedly across most of the reference answers. For example, consider a set of reference answers for a question from the Beetle dataset:

**question:** Explain why you got a voltage reading of 1.5 for terminal 1 and the positive terminal?

**Reference Answers:**
- Terminal 1 and the positive terminal are separated by the gap
- Terminal 1 and the positive terminal are not connected
- Terminal 1 is connected to the negative battery terminal
- Terminal 1 is not separated from the negative battery
4.3. Key Information Extraction (Extractor)

In the example, terms like *Terminal, 1, is, connected* are repeatedly used. Words like *connected* and the similar word *separated* are therefore good candidates for key information. However, words which appear in the question (like *Terminal* and *1*) as well as in the reference answers, are not key information.

Figure 4.2 illustrates the step-wise operation for extracting key information. The steps are:

**Step 1** Stop words and words which appear in the question are removed from the reference answer.

**Step 2** Words with similar meanings are identified. The similarity between each pair of words in reference answers is calculated using Word2Vec (Mikolov et al., 2013b), and the top 25% of most similar word pairs are identified.
Algorithm 1 Algorithm to extract Key Information
1: procedure KISELECT(Question, References, Stopwords, Word2VecSim)
2:   Input: Question is a String
3:   Input: References is a collection of reference answers
4:   Input: Stopwords collection of stop words
5:   Input: Word2VecSim method to calculate word2vec similarity score
6:   Output: List of top 25% KI for the References
7:   for each Word ∈ Question do
8:     if Word ∈ StopWords then
9:       Remove Word
10:    countWord = {} ▶ Dictionary to count frequency of words
11:   for each Reference ∈ References do
12:     for each Word ∈ Reference do
13:       if Word ∈ StopWords or Word ∈ Question then
14:         Remove Word
15:       if Word ∈ countWord then
17:       else
18:         countWord[Word] = 1
19:   for each Reference1 ∈ References do
20:     for each Word1 ∈ Reference1 do
21:       for each Reference2 ∈ References except Reference1 do
22:         for each Word2 ∈ Reference2 do
24:     Sorted_Sim_Scores = Sort(sim, order = descending)
25:     Sorted_Sim_Scores_filter = SelectTop25%(Sorted_Sim_Scores)
26:     countWordScore = {} ▶ Dictionary to track score of words
27:     for each Word ∈ Sorted_Sim_Scores_filter do
28:       if Word ∈ countWordScore then
29:         countWordScore[Word] += 1
30:       else
31:         countWordScore[Word] = 1
32:     for each Word ∈ countWordScore do
34:   return countWordScore

**Step 3** The word pair list from Step 2 is taken as input. The number of times each word appears is counted, indicating how many times a word is used in a similar form.

**Step 4** The original frequency of each word is incremented by the count of same word from Step 3. The frequency of the word after this step indicates its number of usages in similar forms.
The pseudo-code to extract KI is presented in Algorithm 1.

After Step 1, the frequency of words across all reference answers are: \{battery: 3; connected: 2; negative: 2; separated: 2; gap: 1; electrical: 1; different: 1; states: 1\}.

Step 2 then generates word pairs with similarity scores: \{battery, electrical =0.30; separated, connected =0.29; different, separated =0.29; different, negative =0.23; different, connected =0.18; separated, gap =0.16\}.

Step 3 then counts number of times the word occurs across pairs \{separated =3, different =3, connected =2, negative =1, gap =1, electrical =1, battery =1\}.

Finally, Step 4 adds the count to the original frequency of the words, resulting in the order of key information: \{connected: 5; separated: 5; different: 4; battery: 3; negative: 3; gap: 2; electrical: 1; states: 1\}.

This process identifies words with similar meanings that are used repeatedly, and filters out words which occur in the reference answer but do not show similarity with other words (for example battery). This captures the behaviour where each answer is a different way of expressing the same thing.

We use parameter optimisation for selecting the number of key information words from the list. The parameter optimisation is discussed in detail in section 4.5. In section 4.5 we incorporate the key information into the alignment model and measure its empirical effectiveness in the ASAG task.

4.4 Key information model (SACA + KI)

The key information model in our student response grading model is the combination of the key information and an attention model (see Chapter 3 for details of the attention model) as shown in Figure 4.3. The major difference between our attention
model and the key information model is that we add the key information as a feature to the attention model. The attention model provides the alignment between the student response and a given reference answer. The key information then provides the added validation for the correctness of the student response. The key information is generated from the set of reference answers for a question from the Extractor as shown in Section 4.3. To capture the semantic relation of similar words, the key information is embedded into GloVe (Pennington, Socher, and Manning, 2014) vectors. The vectors are encoded using a feed-forward neural network.

The attention model takes as input a reference answer and student response pair and outputs an alignment score. We use the structured variation of the attention model (see Section 3.4.2). Structured attention is suitable for applications that require structural dependencies at the alignment layer. Instead of a single latent variable during alignment, the structured attention model uses multiple latent variables that encode possible structures over the input for alignment (see Section 3.4.2 for a detailed description of the structured attention model). We use syntactic tree selection as these structures showed state-of-the-art results in the ASAG task (Section 3.4.2). These structures are obtained in the model with end-to-end training and without any prior annotation.

As Figure 4.4 shows, our model combines key information with the aggregated word
vectors to the model described in Section 3.4.2, Figure 3.13. The output from the attention model (the aggregation step) is concatenated with the output of the embedded key information and fed into a 3-layer neural network in which each layer contains 600 neurons. Concatenating the output from the attention model and the encoded key information enables the overall model to take both into account when grading student responses. The 3-layer neural network denoted by $Q$ learns the method to combine the key information with the alignment information. Following Equation 3.5, the aggregated vectors are concatenated using:

$$KA = Q([v_1, v_2, KI_1, KI_2])$$

where $KI_1$ and $KI_2$ are the embedding representations of the key information.

The resulting vector is fed through a final classifier $H$, that is a feed forward network followed by a linear layer:

$$\hat{y} = H(KA)$$

where $\hat{y} \in \mathbb{R}^C$ represents the predicted (unnormalized) scores for each class and consequently the predicted class is given by $\hat{y} = \arg \max_i \hat{y}_i$.

The attention part of the network learns the alignment between syntax trees of the reference answer and the student response. The alignment scores and the key information are aggregated through a concatenation step. The concatenation thus yields a representation of the reference answer and the student response that contains both the alignment and the key information. This representation is then used to classify the student response as either correct, incorrect, or partially correct.
Chapter 4. Incorporating Key Information for ASAG

Figure 4.4: Neural network representation of the key information model. The key information is added to the aggregated alignment information.

\[
\tilde{y} = H(KA)
\]

Combining KI
\[
KA = Q([v_1, v_2, KI_1, KI_2])
\]

Figure 4.5: Weighted F1 score for 2-way classification in development set with varying number of KI.
4.5 Evaluation

4.5.1 Evaluation Setup

All the experiments are conducted using the SemEval dataset as described in Section 2.3.

For the SACA + KI model the dataset is tokenised using the Stanford parser (Manning et al., 2014) and uses a 300-dimensional GloVe embedding (Pennington, Socher, and Manning, 2014) to represent words for the input to the neural model. The implementation of all the models uses Torch\(^1\), with a modified implementation of the modular (pluggable) structured attention source code\(^2\). The development set is used for hyper-parameter exploration. We use the top two words from the order list of key information. The model performance was at the highest when using 2 words for key information. The performance dipped on either side, as shown in Figure 4.5 and Figure 4.6 for the 2-way and 5-way classification tasks respectively. The lower scores as the key information increases also show that the key information has a specific role and represents distinct information that can improve model performance. However, the scores are lower when we increase the number of terms used as key information;

---

\(^1\)http://torch.ch/

\(^2\)https://github.com/harvardnlp/struct-attn

![Figure 4.6: Weighted F1 score for 5-way classification in development set with varying number of KI.](image)
this shows that random information cannot be taken as key information. We verify this further using random words as KI in the model (see Section 4.5.2).

Performance of the model is evaluated using weighted average F1 scores. The method presented in Section 4.3 to extract key information requires more than one reference answer in the dataset. Since the SciEntsBank dataset has only one reference answer per question, we could evaluate results on SciEntsBank only with human annotated key information. In these cases, although one can argue that the model sees the key information and the reference answers for both training and testing, the advantage for a critical task like ASAG clearly outweighs the (minor) additional annotation of key information. Key information is needed for each question, so as new questions are added, then along with their reference answers, the expert can add any key information that he or she feels would be appropriate.

### 4.5.2 Results

In Table 4.1 and Table 4.2, we compare our KI model, with the following models:

**SemEval ASAG task top scorer**  We compare with the Feature Engineering model (Sahu and Bhowmick, 2019). The details of the model are given in Section 2.3.1. For Feature Engineering, the authors only provide results for 5-way classification.
4.5. Evaluation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Engineering</td>
<td>0.709</td>
<td>0.6248</td>
<td><strong>0.925</strong></td>
<td>0.658</td>
<td>0.656</td>
</tr>
<tr>
<td>ACA</td>
<td>0.728</td>
<td>0.671</td>
<td>0.662</td>
<td>0.653</td>
<td>0.630</td>
</tr>
<tr>
<td>SACA</td>
<td>0.731</td>
<td>0.674</td>
<td>0.686</td>
<td>0.663</td>
<td><strong>0.661</strong></td>
</tr>
<tr>
<td>Key Information (KI) (machine)</td>
<td>0.728</td>
<td><strong>0.704</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>KI (human)</td>
<td><strong>0.753</strong></td>
<td><strong>0.704</strong></td>
<td>0.715</td>
<td><strong>0.682</strong></td>
<td><strong>0.661</strong></td>
</tr>
<tr>
<td>KI (random)</td>
<td>0.621</td>
<td>0.514</td>
<td>0.521</td>
<td>0.483</td>
<td>0.463</td>
</tr>
</tbody>
</table>

Table 4.2: Weighted F1 for the 5-way classification for KI and other models.

<table>
<thead>
<tr>
<th></th>
<th>Beetle</th>
<th>SciEntsBank</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UA</td>
<td>UQ</td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>2-Way</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KI (machine)</td>
<td>0.86</td>
<td>0.85</td>
</tr>
<tr>
<td>KI (human)</td>
<td>0.86</td>
<td>0.85</td>
</tr>
<tr>
<td>5-way</td>
<td></td>
<td></td>
</tr>
<tr>
<td>KI (machine)</td>
<td>0.74</td>
<td>0.72</td>
</tr>
<tr>
<td>KI (human)</td>
<td>0.77</td>
<td>0.74</td>
</tr>
</tbody>
</table>

Table 4.3: Precision and Recall values for KI(machine) and KI (human) models for 2-way and 5-way classification.

Our previous attention models We compare with our ACA and SACA models. The ACA model is described in Section 3.4.1 and uses word level attention without the structured pattern. The SACA model is described in Section 3.4.2 and uses structured patterns during alignment. We compare with alignment models without key information to investigate the improvement that key information adds.

Table 5.1 and Table 5.2 show that the key information model outperforms both our own attention models and the state-of-art models except Feature Engineering in unseen answers category. This applies to both the machine generated and human annotated KI.

Table 4.3 shows precision and recall for the 2-way and 5-way classification. As in
Chapter 3, we can see that the model mostly shows a balanced precision and recall values while slightly favouring precision over recall in a few categories.

As mentioned in Section 3.5.2 the high score of unseen answers for Feature Engineering is due to reusing correct responses in training. We believe that the human annotation of key information gives the best result due to domain knowledge during annotation. The annotator has access to the student’s response as well, so this creates a bias in selecting key information. For example, in the example presented in Section 4.3, the machine-generated key information was ‘separated’ whereas the human-generated one was ‘gap’. Most of the students’ answers tended to use ‘gap’ rather than ‘separated’.

The structured attention model learns generic features from the grading dataset that apply to new questions as well as previously seen questions. This is bolstered by adding the key information terms which help to capture terms that may not have been seen during training. The practical benefit of this is that the educator does not need to provide a large number of annotated student responses for new set of questions to retrain a model from scratch. For example:

**Reference Answer:** When there is a change in voltage you have found the damaged bulb.

**Student Response (incorrect):** There will be a voltage reading where the bulb is burned out.

The attention models classify this incorrect student response as correct. Those models align change with voltage reading, but change needs to be part of the response for the response to be correct. In other words, the word change is key information. The attention model fails to identify the key information and labels the answer as
4.5. Evaluation

correct. The key information model, however, is provided with change as key information. With this extra information, the model correctly classifies the student response as being incorrect.

This benefit also transfers to Unseen Domains as seen by exceptionally better performance of key information in the unseen questions as seen in table 5.1 and table 5.2. Although the attention models without key information perform better than current state of the art models, adding key information does improve the accuracy notably. This is because, for a model trained to mark student responses on specific topic, the training set usually contains enough information to mark new student responses. However, if new questions (of different domains) are added to the topic, the reference answers to these new questions can contain important domain specific information that the model has not acquired during training. In such cases, providing key information can help the model continue to accurately mark the answers as the key information now incorporates the domain information into the model. This can also be seen in Example 2.6 and Example 2.7. Both the ACA and SACA models could not correctly identify the partially correct responses in the examples. The KI model, however, correctly classified both the responses as partially correct. The use of KI played a vital role where the information could not be obtained due to the lack of training data for alignment models in unseen-question and unseen domains. In Example 2.6 “move” was selected as one of the KI and “surface” and “tension” was selected as KI for Example 2.7.

Figure 4.7 shows confusion matrix for the 5-way classification. The data for the confusion matrix is extracted from the best performing model using KI (human). Comparing the data with ACA and SACA models (Figure 3.21, we can see that KI model shows improve performance in the classification of correct, partially correct, and non-domain data. The confusion matrix for KI model also shows that the number of misclassification for partially correct responses is decreased considerably compared to the ACA and SACA model.
4.6 Conclusion

In this chapter we defined Key Information and also explained its importance for the ASAG task. Additionally, we modelled KI as the common information in a set of reference answers and devised a light-weight algorithm to extract KI. We applied KI successfully in the ASAG task to obtain results that improve on the state-of-the-art. We showed that when combined with alignment models, using the key information yields improved results over previous state-of-the-art models in each of the task categories, including Unseen Question and Unseen Domain. This shows an extra benefit of key information: it can be used to help mark answers where large amounts of labelled student responses are not available for training. Although our model to extract key information requires more than one reference answer, the model that uses key information is still useful for questions with a single reference answer. It is cheaper and easier for a domain expert to add key information along with additional reference
answers, rather than to produce a large set of labelled student responses.
Chapter 5

ASAG Based on Complex-valued Word Embeddings

5.1 Introduction

From a linguistic point of view, we can distinguish between various properties associated with words, such as lexical properties and semantic properties. Some of these properties can be directly observed (for instance, n-grams in case of lexical properties and synonyms and antonyms from a thesaurus in case of semantic properties). But not all properties are directly observable - some are hidden or latent and can only be inferred. Numerous methods are used to extract the latent property of words. For instance, Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan, 2003) models topics as latent properties of words or documents. Similarly, Hidden Markov Model (HMM) can be used for part of speech tagging, where the part of speech is assumed to be a latent property of the words.

In Chapter 4 we showed that from a collection of reference answers we can extract Key Information (KI). Since KI is a property of ASAG data, the next step is to use a neural network for learning KI in an end-to-end architecture; that is, the architecture
should not use KI as an input to the neural network. In this chapter, we use a neural-network architecture that can learn KI and use it for the ASAG task.

For this, we use a technique that uses complex numbers in the word embedding vectors, where the imaginary part of the vector represents the latent property of the words. Li et al. (2018) used complex-valued word representations for the task of binary sentence classification. Li et al.’s method learns latent semantic information about the word, such as sentiment or polarity. The latent information is dependent on the type of dataset. For instance, Li et al.’s model is trained on a sentiment dataset to get the polarity as latent information. The results obtained suggest that the use of complex numbers can indeed catch latent information not accessible to real numbers. We extend Li et al.’s neural network architecture to incorporate two sentences. Since KI is an inherent latent property of ASAG as discussed above, the architecture can then learn KI as latent information. We will also show that the neural architecture can use the learnt KI to perform ASAG based classification.

5.2 Related work

One of the most challenging tasks in Natural Language Processing (NLP) is to capture the meaning of a word or a combination of words, usually expressed in the form of a sentence or a phrase. Currently, the word embedding approaches pioneered in the previous decade (Mikolov et al., 2013a; Pennington, Socher, and Manning, 2014) now form the state-of-the-art with GPT-3 (Brown et al., 2020). GPT-3 represents the meaning of a word in terms of its context, and has successfully augmented many downstream NLP tasks such as text classification, text generation and machine translation.

The fundamental idea behind word embeddings is to use word co-occurrence as the basis of similarity between two words. However, this approach leads to issues when we want to consider word embeddings as a measure of semantic relationship between
words. For example, word embeddings calculate a high similarity between words such as good and bad, or gap and connected because they occur frequently within a small window in text corpora. But the semantics imply that they should have very low similarity values. To overcome these limitations of word vectors, a method proposed by Li et al. (2018) utilises complex-valued word vectors. In these vectors, the imaginary part stores latent semantic information about the word, such as its sentiment or polarity. The complex-valued word vectors also directs how it combines and also directs how it combines with other words to give rise to the meaning of the combination.

In the last few years, the use of complex numbers has been brought to the attention of the machine learning community by papers such as Arjovsky, Shah, and Bengio (2016), Danihelka et al. (2016), Tygert et al. (2016), Jing et al. (2017), Li et al. (2018), Trabelsi et al. (2018), and Mescheder, Nowozin, and Geiger (2018).

Arjovsky, Shah, and Bengio (2016) show that using complex numbers allows the passage of gradient information in the neural network through long sequences to be improved. In their paper, Arjovsky, Shah, and Bengio prove that using complex numbers as the weights of an RNN network has the benefit of reducing the problem of vanishing gradients. In a similar line of research, Jing et al. (2017) used complex numbers to better capture long distance dependencies and to provide a more robust forgetting mechanism. Danihelka et al. (2016) used complex-valued vectors to enlarge the memory of an LSTM without increasing the number of parameters. Their experiments show the use of complex numbers yields higher memory capacity networks.

Trabelsi et al. (2018) do not limit the use of complex numbers to RNN but explore their application for convolution networks. Trabelsi et al. prove that the use of complex values allow to for more competitive results to be reached in vision tasks. The advantage of using complex values was also presented by Mescheder, Nowozin, and
Chapter 5. ASAG Based on Complex-valued Word Embeddings

Geiger (2018) who applied them for training Generative adversarial network (GAN). The use of complex numbers stabilised the training process. Additionally, it tackled mode collapse, a common training issue in GAN. Mode collapse is a failure of a GAN network to produce variety in its outputs.

Complex-valued word representations were used by Li et al. (2018) for the task of binary sentence classification. The results suggest that complex-valued vectors can capture more semantic information than the real-valued vector representations.

Similar complex-valued word representations were also used by Li, Wang, and Melucci (2019) for a matching task in a neural network setting. The neural architecture used in this Chapter is related to the work of Li, Wang, and Melucci (2019). To the best of our knowledge, this is the first time the complex-valued word representation has been used in the ASAG task. More importantly, this research shows how complex valued word representation can support the idea of key information.

5.3 Methodology

In this section we describe how we can model text in the form of complex numbers, and exploit the interactions permitted by this representation. First, we describe word embeddings in the form of complex numbers. We then present a method to use this word embedding to calculate the semantic relationship between sentences. As mentioned in the earlier section, the complex number representation gives the semantic representation of a word along with any latent properties. We then present a neural network architecture that can use the embeddings with complex numbers for the ASAG task.

5.3.1 Complex Word Embeddings

A complex word embedding is formed using a mixture of GloVe lookup (Pennington, Socher, and Manning, 2014) and phase lookup. The GloVe lookup gives the real part
of the complex embedding and the phase lookup gives the imaginary part. The real part gives the distributional semantics (Turney and Pantel, 2010) of the word and the imaginary part corresponds to a latent concept for the word.

A word can be considered as a superposition state of all its latent concepts and the context of its usage determines which concept or concepts are the most important. For example, in the case of the ASAG task, the complex vectors can represent the key information required for the student response to be correct. In such a scenario, the latent concepts within the complex vectors represent the degree of importance of the words for the particular answer. Then its linear combination is its different importance level. In the complex word embeddings approach, words are represented in a Hilbert Space, which is a complex vector with the inner product of two vectors defined as:

\[
|w\rangle = \sum_{j=1}^{n} r_j e^{i \phi_j} |e_j\rangle
\]

where \(|w\rangle\) represents a word vector and \(i\) is the imaginary number such that \(i^2 = -1\).

In equation 5.1, \(r_{j=1}^{n}\) are non-negative real-valued numbers satisfying

\[
\sum_{j=1}^{n} r_j^2 = 1
\]

and \(\phi_j\) are the corresponding complex valued phases.

5.3.2 Density Matrix Representation of Words

Once the complex embeddings of each word in a sentence are formed, they are summed to form a density matrix. The complex-valued off-diagonal elements in the
density matrix describe the correlations between the words’ dimension, while the diagonal entries correspond to a standard probability distribution. The off-diagonal elements provide the framework to model the possible interactions between the words. The formation of a density matrix gives us a mechanism to correctly model word interaction, such as the interaction between negation and a phrase like not connected. The combination of words is computed as:

\[
\rho = \sum_{j} \frac{1}{m} \vert w_j \rangle \langle w_j \vert
\]  

(5.2)

where \( m \) is the number of words. Equation 5.2 produces a density matrix \( \rho \) for the composition of words. The complex-valued off-diagonal elements describe the correlations between words’ dimension, while the diagonal entries correspond to a probability distribution. The off-diagonal elements provide the framework for potentials to model the possible interactions between the words.

5.3.3 Interaction Between Density Matrices

Since the density matrix represents a sentence, the interaction between two density matrices gives us a measurement between two sentences. In the case of ASAG, this measurement can give us the required information to grade the student response. The interaction is measured using Von Neumann (VN) divergence (Sordoni, Nie, and Bengio, 2013). The interaction is comparable to the alignment model in Chapter 3. The alignment, however in this context, is between the real part and the imaginary part respectively of each sentence. For two sentence density matrices \( \rho_1 \) and \( \rho_2 \), the VN divergence is defined as:

\[
\Delta_{VN} = -tr(\rho_1 \log \rho_2)
\]  

(5.3)
5.3. Methodology

5.3.4 Neural Network for ASAG Using Complex Embeddings

As shown in Figure 5.1 the input to the system is a reference answer and student response pair. Each of the answers and responses is modelled as a combination of words. For each word GloVe vectors and phase vectors are generated. The glove vectors are extracted from GloVe (Pennington, Socher, and Manning, 2014) word embeddings. The phase vectors are generated from a randomly initialised lookup. The phase lookup is set to trainable, thus its value changes as the model learn from the data. The GloVe vectors and phase vectors of the reference answer and student response are mixed to form complex embeddings using Equation 5.1. The complex embedding of a reference answer and a student response is then converted into density matrices using Equation 5.2. The matching between the two density matrices is calculated using Von Neumann Divergence using Equation 5.3.
Figure 5.2: Multi-Tasking Learning model that learns response classification and KI tagging.
5.3. Methodology

5.3.5 A Baseline Comparison Based on Multi-Tasking Learning (MTL-ASAG)

The complex number method learns key information as latent information. To compare the effectiveness of the learnt key information we compare the complex number model with a baseline system that uses human labelled key information to learn key information from the data. In order to compare such the baseline model with a complex number model, to devise the baseline model can output both key information and student response classification. The baseline model will jointly learn both the key information and response classification. For joint learning we use Multi-Tasking Learning (MTL). MTL focuses on learning a model for one task better, by using knowledge contained on a sub-set of tasks (Zhang and Yang, 2021).

We use the student response classification and the learning of KI as our two subsets of the tasks. We use MTL with transformer-based models (BERT) (Devlin et al., 2018).

The architecture of the MTL model is shown in Figure 5.2. The BERT pre-trained layer is shared by both the tasks. The input is the pair of reference answer and the student response separated by special token [SEP]. The combined length of the pair of the sentences is M. The start is indicated by a special token [CLS], and the end is indicated again by [SEP]. The BERT model first converts the input sequence to a sequence of embeddings, and then uses multiple attention layers to extract contextual information. The contextual information is present in each token $(t_1, ..., t_m)$ and the combined contextual information is present in the output [CLS] token. We use the output [CLS] token for the response classification and the each individual token for tagging KI.

The contextual information from BERT is used by the task-specific fully connected layer. For response classification, we use softmax to classify into the 2-way or 5-way classification and use categorical cross-entropy loss to train the model. For tagging
KI information, we tag each individual token as either KI or not KI. We use softmax to classify the token and again use categorical cross-entropy loss to train the model.

The MTL model is trained in a similar way to the one used by Tu et al. (2020). We use a BERT model pre-trained using a masked language modeling technique (Devlin et al., 2018). The task-specific layer is randomly initialised. In each epoch a mini-batch of the dataset is used, and then the model is updated according to the task specific objective. The parameters are learned using mini-batch based stochastic gradient descent.

## 5.4 Experiments

### 5.4.1 Method: Evaluation Setup

All the experiments are conducted using the SemEval dataset as described in Section 2.3. The dataset is tokenised using the Stanford parser (Manning et al., 2014) and uses a 50-dimensional GloVe embedding (Pennington, Socher, and Manning, 2014) to represent words as GloVe vectors. The values of the lookup table for the complex vectors are randomly initialised under the normal distribution of \([-\pi, \pi]\). We use the categorical cross-entropy loss. The embedding layer of the complex vector is set to trainable. We used unseen answers as the development set and unseen questions as a test set and vice-versa. For unseen domains, we used both unseen answers and unseen questions as the development set. The development set in each instance is used for selecting the best model. The best result in the development set is found with the parameters 50 hidden units, 0.7 dropout keep probability, l2 loss, a batch size of 32, a learning rate of 0.05 and the Adagrad optimiser algorithm (Duchi, Hazan, and Singer, 2011). Performance of the model is evaluated using weighted average F1 scores.

For the MTL-ASAG model we followed the parameters used by Tu et al. (2020). We
5.4. Experiments

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ACA</td>
<td>0.826</td>
<td>0.802</td>
<td>0.783</td>
<td>0.745</td>
<td>0.701</td>
</tr>
<tr>
<td>SACA</td>
<td>0.831</td>
<td>0.810</td>
<td>0.791</td>
<td>0.758</td>
<td>0.725</td>
</tr>
<tr>
<td>SACA + KI</td>
<td>0.852</td>
<td>0.835</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MTL-ASAG</td>
<td>0.806</td>
<td>0.783</td>
<td>0.758</td>
<td>0.675</td>
<td>0.682</td>
</tr>
<tr>
<td>Complex Embedding</td>
<td>0.831</td>
<td>0.840</td>
<td>0.801</td>
<td>0.792</td>
<td>0.742</td>
</tr>
</tbody>
</table>

**Table 5.1:** Overall Accuracy for the 2-way classification for complex embedding and other models.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ACA</td>
<td>0.728</td>
<td>0.671</td>
<td>0.662</td>
<td>0.653</td>
<td>0.630</td>
</tr>
<tr>
<td>SACA</td>
<td>0.731</td>
<td>0.674</td>
<td>0.686</td>
<td>0.663</td>
<td>0.661</td>
</tr>
<tr>
<td>SACA + KI</td>
<td>0.728</td>
<td><strong>0.704</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MTL-ASAG</td>
<td>0.728</td>
<td>0.629</td>
<td>0.631</td>
<td>0.586</td>
<td>0.630</td>
</tr>
<tr>
<td>Complex Embedding</td>
<td><strong>0.742</strong></td>
<td>0.686</td>
<td><strong>0.686</strong></td>
<td>0.671</td>
<td>0.656</td>
</tr>
</tbody>
</table>

**Table 5.2:** Overall Accuracy for the 5-way classification complex embedding and other models.

use the BERT base model for our experiment. The human annotated KI data is used as the labelled data for the training and testing.

In Table 5.1 and Table 5.2, we compare our complex vector neural network model, with the baseline MTL model and a Key information (SACA + KI) model that uses key information combined with structured attention. The SACA + KI model is described in detail in Chapter 4.

5.4.2 Results

From Table 5.1 and 5.2 we can see that the complex embedding model outperforms all the other models except the key information model in both 2-way and 5-way task. Table 5.3 show precision and recall value for the complex embedding model.
Chapter 5. ASAG Based on Complex-valued Word Embeddings

<table>
<thead>
<tr>
<th></th>
<th>Beetle</th>
<th>SciEntsBank</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UA</td>
<td>UQ</td>
</tr>
<tr>
<td>P</td>
<td>R</td>
<td>P</td>
</tr>
<tr>
<td>Complex Embedding</td>
<td>0.85</td>
<td>0.82</td>
</tr>
</tbody>
</table>

|        | Beetle  | SciEntsBank |
|        | UA  | UQ  | UA  | UQ  | UD  |
| P     | R   | P   | R   | P   | R   |
| Complex Embedding | 0.75 | 0.73 | 0.68 | 0.68 | 0.68 | 0.69 | 0.67 | 0.67 | 0.67 | 0.65 |

Table 5.3: Precision and Recall values for Complex Embedding models for 2-way and 5-way classification.

It also outperforms the KI model in the unseen answer category for the 5-way task. The improvement over other models, especially the strong alignment models from Chapter 3, shows that using the imaginary number to encode latent information helps improve the alignment models. Since the KI model improved upon the alignment model, and from our description of latent information for ASAG in Section 5.1, we can indicate that the complex embedding model learns key information from the ASAG data.

The performance of the baseline MTL-ASAG model is comparable to previous state-of-the-art models but is worse than the complex embedding and non-random KI models. From the MTL-ASAG model we get two different results. The results on the response classification are presented in Table 5.1 and Table 5.2. Additionally, the MTL model also produces KI tagging.

The MTL model performs very poorly on the KI tagging task, with a F-score of 0.431 and 3.86 KI per sentence. Since the MTL model shares parameters for both the tasks, the low F-score on KI tagging task brings down the F-score on the response classification task. We also performed a further experiment by switching off the KI tagging task and using the same MTL model to perform only response tagging task. This experiment used the same training parameters as MTL-ASAG model but was trained only on a single classification task without using the KI tagging task. The results of this model are shown in Table 5.1 and Table 5.2 as the BERT-ASAG model.
We can see that the BERT-ASAG model gives the best result in the unseen answers category and competitive results in the other categories. The result for the BERT-ASAG model is in tune with the recent success of BERT model in other NLP tasks, including open-domain question answering, FAQ retrieval, and dialogue generation (Wang et al., 2019; Sakata et al., 2019; Kim et al., 2020).

We can conclude two important findings from the MTL experiments. The first is that instead of using KI tagging as secondary task, if we use KI directly in the model, as in KI models or as latent information in a complex embedding model, we get better results. This shows that complex embedding method is a better way to use the KI information as latent information and train under only one loss function compared to two separate loss function for the MTL-ASAG method. The second finding is that KI is an important aspect of ASAG as using KI we can get better results than a very strong BERT base model.

Additionally, we also saw that the KI model has better performance in unseen questions and unseen domains category than other listed models. This trend also continues with the complex embedding model as seen in Table 5.1 and 5.2. For the KI model we attribute the improvement in performance to the question and domain-specific information that KI carries. The complex embedding model is generic in grading the student responses but when ported to a new question category and a domain, the complex embedding model requires the additional domain-specific knowledge that KI holds for accurate classification. This further shows that the latent information learnt from the ASAG dataset by complex embedding model is the key information.
5.5 Conclusion

In this chapter, we introduced a system to perform the ASAG task through the use of complex-valued words representations. Our preliminary results show that complex-valued word representations learn latent information corresponding to the dataset. In the case of ASAG, the latent information is word combination and key-information.
Chapter 6

Conclusion

In this dissertation I focused on an important task in natural language processing: Automatic Short Answer Grading (ASAG). The general approach in the dissertation has been to compute the semantic similarity between the student response and a provided reference answer. The semantic similarity is computed using word-level alignment and structural alignment. I also presented the idea of Key Information as an important element of automatic grading. The initial alignment-based approach was improved when augmented with KI. In this concluding chapter, I summarise the research questions, major findings, and contributions of this work, and point out its limitations, leading to the important discussion of possible future work.

6.1 Research Questions and Findings Revisited

Research Question 1: How can we use the concept of semantic similarity in the ASAG task?

Given an expert-provided reference answer to a short-answer question, correct student responses should be semantically similar to the reference answer. Based on this hypothesis, I have developed an automatic short answer grader (Section 3.4.1) that uses a successful neural-based semantic similarity approach to compute the semantic similarity between student responses and reference answers. The neural model uses
the concept of alignment based on semantic similarity and is augmented with an aggregation step to automatically grade short answers. The neural model demonstrates top result in the SemEval dataset.

**Research Question 2: Is the linguistic information encoded within syntactic structure an important signal for ASAG?**

Motivated by the successful use of structure in related semantic similarity task (Section 2.2.3.1), I used a modified version of word-based alignment neural network to encode syntactic structures and its comparisons (Section 3.4.2). The neural network automatically generated the structures from reference answers and student responses and used the generated structures for alignment. The improved results of the structure-based model over the word-based model show that the syntactic structure is an important signal for ASAG.

**Research Question 3: How can we identify the most important information in an answer to a question, and use it to help decide whether a student response is correct?**

In Chapter 4, we define Key Information (KI) as the new information that is not present in the question and which is necessary for the answer/response to be correct. Based on this hypothesis I have devised Algorithm 1 to select KI from reference answers. To evaluate the extracted KI I have developed a neural network model that uses structural alignment along with KI for ASAG task. The model demonstrates top results compared to the alignment models and performs especially well in the unseen domain category. As a next step, we hypothesise that KI is a latent property of answers and thus learn this latent property as embeddings using complex numbers (Section 5.3.4). The model shows motivating results in classifying student responses.
6.2 Limitation

This work raises two important points which I will discuss in this section. First of these is the reliability of an automatic assessment system. It is an important question whether the numbers for the ASAG systems are good enough for real life marking. The inter-annotator agreement on the SemEval ASAG dataset is 86% (Dzikovska et al., 2013a) for the five-way classification. This shows that humans can reach a consistency of 86% in agreeing on the grades. Although we achieved state-of-the-art results for the 5-way classification in the popular ASAG dataset, we are not at the 86% mark. This suggests that the automatic approaches on their own are still not completely ready for a critical task such as response grading, particularly in a high stakes setting. Having said that, automatic grading can have a very important role, especially for providing feedback through the learning period of a student. The rapid feedback can take the pressure off the teachers, who can then concentrate more on developing the learning materials. Additionally, the quick feedback can also provide vital information to students on their strengths and weaknesses, thus enabling students to take better control of their own learning. Therefore, automatic grading is still not at a stage where it can be relied upon for critical examinations but could be used in conjunction with a human involved in the loop.

Second, ASAG research is greatly inhibited by the lack of a reliable and big dataset. Although neural networks are very powerful and have state-of-the-art results in many natural processing tasks, they need a huge corpus to produce accurate results. Currently, the publicly available ASAG datasets are not large enough to produce comparable results to similar semantic similarity problems. For instance, a similar alignment model used in this dissertation (Chapter 3) produces accuracy of 88% in the 5-way language inference task (Parikh et al., 2016). That is more than 15% improvement over our best result in ASAG. The size of the language inference corpus (Bowman et al., 2015) is more than ten times the size of the ASAG corpus used in this dissertation. Additionally, ASAG datasets can suffer from low inter-annotator
agreement. This is due to the inconsistency in marking. For instance, for a question "What is a travelling salesman problem?" and for a reference answer: "Visit every location only once using the shortest path" the student response “Visit every location using the shortest path” has been marked as correct, incorrect, and partially-correct by three different markers in the marking cohort\(^1\). This discrepancy in marking can be the result of various conditions including non-concise marking guidelines and the mood of the markers (Townsend, Kek, and Tuck, 1989).

### 6.3 Future Directions

There are many directions that the research presented in this dissertation can move in the future. To start, we can use more sophisticated neural network models that can show better performance for ASAG. The work presented in this dissertation have showed that we can obtain state-of-the-art results using ACA and SACA models. Additionally, the current models can also improve by using the now popular and effective embedding models like ELMo (Peters et al., 2018), BERT (Devlin et al., 2018), GPT-3 (Brown et al., 2020).

The dissertation shows the importance of KI in ASAG. The dissertation also presents two separate methods (Chapter 4 and 5) to extract KI. To the best of our knowledge this is the first definition and use of KI. One possible future direction will be to analyse the complex embedding model in more detail and understand the role taken by the imaginary numbers to extract KI. Given the initial stage of our proposal for complex embedding, several research directions can be investigated. Firstly, the architecture of the system we propose can be enriched by the use of more sophisticated models. Secondly, a large scale human experiment should be performed to understand the role played by the imaginary numbers to capture word meaning. Such experiments

\(^1\)This example was seen in a real university exam. We cannot cite it due to the confidentiality agreement.
should provide useful information about the use we can make of the complex-valued vectors. Finally, the power of the vectors we trained can be tested in other NLP tasks, such as the automatic evaluation of Natural Language Generation (NLG) systems.

I believe that the most important direction is the pursuit towards explainability of ASAG models. Although we did touch the explainability in Chapter 3, the explainability came only from one part of the architecture. It would be highly beneficial for the improvement of the ASAG model if we can exactly explain why the model came to a specific conclusion. The dissertation lays a foundation for explainability by using heat-maps to show word level relation during response classification. It is also important to note that the human-added key information adds an element of explainability. Therefore, it will be equally important to pursue methods to build upon the existing success of the human ability to add key information. The explainability would certainly help us rely more on automatic grading even in critical exams scenarios and will move the ASAG research in a more reliable and accurate direction.


Dzikovska, Myroslava O et al. (2013a). *Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge*. Tech. rep. NORTH TEXAS STATE UNIV DENTON.


Islam, Aminul and Diana Inkpen (2008). “Semantic text similarity using corpus-based word similarity and string similarity”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 2.2, pp. 1–25.


Jing, Li et al. (June 2017). “Gated Orthogonal Recurrent Units: On Learning to Forget”. In: *Neural Computation* 31. DOI: 10.1162/neco_a_01174.


Mitchell, Tom et al. (2002). “Towards robust computerised marking of free-text responses”. In:


Pepple, Dagogo J, Lauriann E Young, and Robert G Carroll (2010). “A comparison of student performance in multiple-choice and long essay questions in the MBBS stage I physiology examination at the University of the West Indies (Mona Campus)”. In: Advances in physiology education 34.2, pp. 86–89.


Ramachandran, Lakshmi, Jian Cheng, and Peter W. Foltz (2015). “Identifying Patterns For Short Answer Scoring Using Graph-based Lexico-Semantic Text Matching”. In: BEA@NAACL-HLT.


Bibliography


Turney, Peter and Patrick Pantel (2010). “From frequency to meaning: Vector space models of semantics”. In: Journal of Artificial Intelligence Research 37, pp. 141–188.


Woolf, Beverly Park et al. (2013). “AI grand challenges for education”. In: *AI magazine* 34.4, pp. 66–84.


Yao, Xuchen et al. (2013). “Answer extraction as sequence tagging with tree edit distance”. In: In North American Chapter of the Association for Computational Linguistics (NAACL.


Yu, Dong et al. (2011). “Introduction to the special section on deep learning for speech and language processing”. In: IEEE Transactions on Audio, Speech, and Language Processing 20.1, pp. 4–6.

