Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

Automatic Short Answer Grading in an Online Quiz System

John Faucett

Master Thesis

Betreuer: Prof. Dr. Jörg Lohscheller

Trier, 11.01.2019

Preface

First and foremost I would like to thank my wife Anna for supporting me in this entire endeveour and without whom none of this would have been possible. I would like to thank Prof. Dr. Lohscheller for never wavering to provide his time, insightful feedback and critical thoughts at every stage along the way. And finally, I would like to thank my brother Nathan for all his work in helping to build Stembord over the years.

Abstract

This thesis researches the problem of Automatic Short Answer Grading (ASAG) in order to build an automatic grading module for short-text responses. This module is developed for an online Learning Management System (LMS) called Stembord. A Multi-layer Perceptron model is the best performing model based on the features obtained during research. This research also informs the design and implementation of the short-answer grading module which is consequently integrated into an API service and called by the Stembord application.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit dem Thema Automatic Short Answer Grading (ASAG) um ein automatisches Benotungsmodul für Freitext Fragen zu konstruieren. Das Modul ist für ein bestehendes online Learning Management System (LMS) namens Stembord entwickelt. In der Forschung, stellt sich heraus dass, basierend auf Merkmale aus dieser Forschung, ein Multilayer-Perceptron das beste Modell ist um Freitext Fragen automatisiert zu benoten. Diese Forschung prägt das Design und die Entwicklung vom Benotungsmodul für Freitext Fragen, welches nachfolgend als externe API gebaut und von der Stembord Application abgerufen wird.

Contents

1

Introduction

> "It is a capital mistake to theorize before one has data."
> — Arthur Conan Doyle, A Study in Scarlet , Part 1, chap. 3

This thesis describes the construction of an Automatic Short Answer Grading (ASAG) module in an online learning platform known as Stembord. It begins with an introduction to subjects important to the task, which are Automatic Short Answer Grading, Intelligent Tutoring Systems (ITS) and Online Learning Management Systems (LMS). After this necessary background knowledge has been introduced, the intersection of these domains is delineated in order to elucidate the set of specific problems which arise when researching and developing an ASAG module for a particular LMS known as Stembord.

The main goal of this thesis is to research ASAG and to design and integrate an ASAG module derived from that research into an online quiz system. Ideally, the solution would be state-of-the art, however, this is not the primary goal. The primary goals are to research and understand the problem domain, develop a software infrastructure designed for long-term sustainability and incremental improvement of the developed automatic grading module and to carve out key insights for directing future development of the short-answer grading module.

There are many complex domains which need to be understood in order to evaluate a software system designed for Automatic Short Answer Grading. In the proceeding sections, these areas, as well as fundamental problems and issues within the field of ASAG itself, will be examined in detail.

1.1 Automatic Short Answer Grading (ASAG)

1.1.1 Structure and Features

Automatic Short Answer Grading or ASAG is made up of keywords which can be used to understand the subcomponents of the discipline. *Answer Grading* encompasses the set of activities whereby a teacher assigns numerical grades to student responses for various question prompts, exams or tasks within a classroom context. It is a subset of tasks within the more general field of student assessment, which, importantly, is one of the teacher's most complex and demanding tasks [CF17].

The assessment of students' progress, typically referred to as learning outcomes, can be facilitated by a plethora of question types and divergent grading methods [BGS14]. *Short Answer* refers specifically to the question type, and although it may be effortless to distinguish between *short-answer* and *multiple-choice* question types, the lines become fuzzy if one compares a *short-answer* response to an essay response. To help alleviate this problem of ambiguity, Burrows et al. define five criteria which can be used to characterize *short-answer* questions and distinguish them from other types of free-text responses such as student essays or *fill-in-the-blank* style questions [BGS14]. These criteria are:

1. The question response must recall external knowledge instead of requiring the answer to be recognized from within the question prompt.
2. The question response must be in natural language.
3. The question response length should be roughly between one phrase and one paragraph.
4. Assessment of the response should focus on content as opposed to writing style.
5. The level of openness in responses should be restricted by objective question design.

These criteria define a *short-answer* as a natural language student response of maximally one paragraph in length, in which the student must recall already learned knowledge without help from texts and which is constrained in its semantic scope by the specificity of the question design.

Fig. 1.1 shows a prototypical example of a *short-answer* response for illustrative purposes. The components are taken from the widely used Mohler and Mihalcea dataset, which is also the main dataset used throughout this thesis [MBM11].

Several interesting aspects of ASAG are illustrated in Fig. 1.1. First, the student response contains the pronoun *it* which implicitly refers to *function signature* even though the phrase *function signature* never occurs in the student response. This demonstrates the fact that an extensive amount of knowledge and meaning is implicit in *short-answer* responses. In this case, that implicit knowledge can be found in the question prompt, but that is not always the case. Also the phrase *types of* has been entered incorrectly by the student as *typesof*. This is indicative of another problem in ASAG - dealing with input noise, misspellings, grammatical mistakes and confusing wordings. These problems as well as many others are revisited in later chapters in more detail.

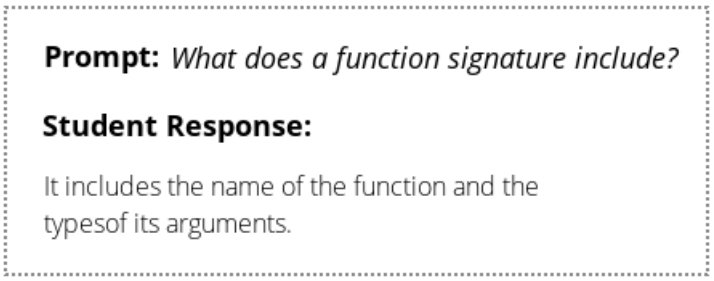

Fig. 1.1. Example of short-answer prompt and student response question item. Notice the spelling error and the pronoun *it* which references *function signature* in the question prompt.

Finally, there is the *automatic* portion of ASAG, which refers to the mechanical process, carried out by a machine, or, according to Burrows et al., "completed using computational methods" [BGS14]. However, this implicit yet normative claim underlying the entire discipline, namely, that assessing student responses should be automated, has practical and philosophical considerations which need to be examined and are, therefore, analyzed in the next section.

1.1.2 Importance and Associated Risks

There are arguments for and against ASAG. These arguments are usually framed and expressed in the context of automated assessment in general, but since ASAG is a specific subfield within that more capacious context, the arguments remain valid and are presented below.

Arguments for ASAG

There are numerous benefits of automatic grading in general and specifically for the automated grading of natural language responses. [BGS14] From the teachers perspective, the task of student assessment can be seen as a monotonous, repetitive and time consuming job which demands a great deal of overhead and is non-rewarding [RND15]. Compared to other question types, the task of short-answer construction is also more economic, since constructing high-quality multiple-choice items, for instance, is time intensive [Hea00]. Further, time spent on grading does not often advance the teachers knowledge of the subject matter [RND15]. In the same vein, it can also be seen as taking time away from the teachers main task which is to teach the students certain materials. This sentiment is best exemplified by one teacher who said, "It's hard to focus on your students' needs ... when your view is obscured by piles of marking." [Pea18].

Furthermore and in general, assessment of short free-text answers although in many contexts more beneficial to students than multiple choice questions is much more laborious and subjective for the teacher and consequently possibly under-applied [RND15]. Further, multiple-choice items lend themselves to test taking strategies, which do not necessarily evaluate student comprehension as well as

short-answer responses [Hea00]. Clarifying some of the cognitive benefits of short-text responses, Jordan and Mitchell have the following to say in their research:

> "Short-answer constructed response items require the respondent to construct a response in natural language and to do so without the benefit of any prompts in the question. This implies a different form of cognitive processing and memory retrieval when compared with selected response items." ([JM09])

Then there is also the scalability dimension. The demands of large class sizes, especially due to the emergence of Massive Open Online Courses (MOOCs), and the growing number of students, curricula and exams, necessitate some form of automation if only to keep pace with growing demands. Another driving and tandem force is the need for cost-effective and efficient solutions [GHW12].

From a students perspective, answering short-answer questions is more like answering real-world questions than checking multiple-choice items [Hea00]. There is also the immediacy that automatic grading systems provide. Under normal conditions, test-takers must wait for a human to grade their exams yet an automated system can provide instant feedback [BGS14]. Not only can feedback be provided in real-time but the availability of quizzes and exams is independent of teacher presence which means learning assessments are accessible on the student's schedule allowing for more flexible learning as well as more total practice hours, for instance, when quizzes can be repeated [JT16].

Then, there is also the role that human subjectivity and bias plays in grading. Inter-grader agreement between different teachers is not $r \approx 1.0$. This means that given any two human graders students will receive different grades on the same question item by those graders. There are many disparate variables which are associated with this for short-answer responses, some of these are normal mistakes, teacher fatigue, psychological biases, teacher preferences for or biases against certain students or even ordering effects [BGS14].

Automatic grading of short-answer responses can remove or alleviate many of these problems, so an optimistic view would conclude that researchers should barrel along tackling the problem head on while striving for complete automation.

These are all practical arguments in an educational context which one can give as the case for ASAG. However, from a research perspective, ASAG sits at a junction with many other complicated and important subtasks in natural language processing and is considered a hard problem [RND15]. Therefore, improvements in ASAG could potentially provide insights into many other related and important tasks, many of them outside of automated assessment, such as paraphrase recognition, Short Text Similarity (STS) or Recognizing Textual Entailment (RTE) to name a few.

Arguments against ASAG

The arguments against automated short-answer assessment are numerous but none-the-less important to take into account for those who desire to construct

automated grading systems since the starkest critiques can provide key insights, illuminate weaknesses and thus aid in improving hypothesized or established models.

First, there is a broad case against technology in the classroom which states that students may not have equal access to technological resources and, therefore, technological systems could be biased against those students and create barriers for them [Hyn18]. Given the broad distribution of learning tasks and student abilities, adding technology may not always be the optimal instructional method and depending on the technology it may also distract students and even hinder learning [Hyn18].

Then there is the attack against grading itself, known as degrading. This is a slew of arguments backed by research which states that grades themselves reduce students' preference for challenging tasks, reduce students' interest in learning itself, reduce quality of students' thinking, distort curricula, encourage cheating, waste lots of teacher time which could be better spent on teaching and ruin student-student relationships as well as student-teacher relationships [Koh99]. Worse still, this strain of criticisms asserts that grades themselves aren't valid, reliable or objective [Koh99]. This last objection is worth further inspection, since this thesis is about constructing a grading module and if grades are completely unreliable that begs the question as to the point of the entire endeavor.

Kohn builds his case that grades are highly subjective by citing research which shows that any given assignment may well be given two different grades by two equally qualified teachers, and that furthermore, it may be given different grades by a single teacher who reads it at two different times [Koh99]. Moreover, selection bias can play a significant role as teachers decide what particular skills to assess and importantly which questions to leave out as well as how many points each topic is worth [Koh99].

Synthesis

The majority of arguments against automated assessment thus boil down to critiques either of technology or grading itself. These critiques are numerous, informed and in many situations make a powerful case. The task now is to assess them.

First, the arguments against technology are, in the author's opinion in some sense, valid. Technology should not be approached as the only tool which can solve every learning task and, in those cases where technology, although it may provide for better outcomes for the most students, is not equitably available for all, then teachers should certainly take this information into account when developing curricula. However, this critique is irrelevant when it comes to what one's policy should be towards educational technology research and innovation. Given that technology is just another term for a set of available tools designed to help solve a specific problem, a society which can provide more and superior technology i.e. tools built to solve learning difficulties in specific tasks, will, at the very least, have a superior ability to cover the populations' learning needs distribution than societies without such tools. This seems to be a desirable quality whether or not one

consequently decides to actually use the technology in the classroom - having the option is certainly better than not having it. Therefore, one can at least conclude that as it pertains to the technology critique this research endeavor is still valid one.

As for degrading arguments, even Kohn in his essays on degrading freely admits that performance measurements and feedback mechanisms are needed and even desired in the classroom. This indicates occasion for measuring in a quantitative manner the performance of students, which is the ideal if not the reality of grading itself [Koh99]. Further, many of the criticisms essayed against grading fall short when viewed in the context of automated assessment. Special among those is the grader bias. A digital system can choose not to display student names to human graders, can randomize sorting and disallow a human grader to grade many answers in a single time period thus eliminating some of the prevalent human biases during grading. Moreover, it can mechanize and standardize the process and by removing much of the human element, hypothetically eliminate many of the subjective forms of grading bias.

Finally, there is the practical consideration that currently virtually all educational systems mandate the use of grading throughout their infrastructure and rather than concentrating on developing a new alternative that alleviates many of the valid issues raised by degrading, in the author's opinion research should utilize the criticisms to focus on specific areas which, regardless of whether grading is applied or not, are beneficial in a learning context. This latter approach appears to be the most flexible and future proof.

So what is the synthesized conclusion? The specific areas in which automated assessment systems should, in the author's opinion, focus are outlined below.

1. Quantitative performance measurements whose numbers are transparent and easily interpretable by humans.
2. Clear assessment standards and rubrics, which are easily interpretable by all users of the system
3. Informative feedback mechanisms
4. Focus on the human psychological element, both to mitigate biases and to encourage positive learning behavior.
5. A view of technology as an aid not as a end-all solution.

If any automatic grading system focuses on these underlying characteristics then it should be beneficial irrespective of whether the final performance measurements are scaled and tallied to provide a final student grade or used only to help improve learning outcomes in a degrading school.

1.1.3 History and Place in the Learning Technology Landscape

The evaluation of short-text responses overlaps with many disparate domains ranging from Computer Assisted Assessment (CAA), Natural Language Processing (NLP), Intelligent Tutoring Systems (ITS), to Online Learning Management Systems (LMS). However, the details of each intersection between ASAG and a related

domain are not all of equal importance for the purposes of this thesis. For this reason a focus is made on the historical development of ASAG as a subfield within Automated Assessment and Natural Language Understanding (NLU). ASAG's relationship with ITS and LMS is clarified later as those concepts are visited in the context of Stembord, an Online Learning Management System developed by the author.

Historical Eras of ASAG

The very first systems for processing free text responses were developed in the 1960s. One of the first published articles on the topic was an essay by Ellis Page in which he contemplated the inevitability of essay grading by computer and attempted to persuade educators of the merits of such a system and the progress being made in the field, all in 1966 [Pag66]. Then and up until now the problem of automatically processing and grading free text responses has been concentrated into two main subtasks: (1) essay grading and (2) grading of short free-text answers [RND15].

Research has made much progress on essay grading tasks since statistical methods tend to work well there and less semantic understanding of textual content is necessary to build robust solutions than is the case in short-answer grading [RND15]. However, progress has been made in ASAG as well and a number of commercial and non-commercial software systems have been built which provide automated short-answer grading, with the venerable C-Rater from ETS being one of the most notable [BGS14]. Overall, there has been much research and development within ASAG, as Burrows et al. discovered 35 ASAG systems and two competitions in their survey of ASAG.

Burrows et al. identify five areas which are useful for conceptualizing the historical development of ASAG and are reiterated here because each resolves and elucidates specific problems within the research domain. They are:

1. The Era of Concept Mapping
2. The Era of Information Extraction
3. The Era of Corpus Based Methods
4. The Era of Machine Learning
5. The Era of Evaluation

The Era of Concept Mapping

In concept mapping, student responses are viewed as a collection of concepts and the goal of the system is to detect the presence or absence of each concept during the grading process. Concepts can be viewed at a *sentence level* or in what is known as *facet mapping* at a phrase level [BGS14]. Sentence level concepts could be "GDP and life expectancy are positively correlated" or "Claudius is the King of Denmark in the play Hamlet" whereas facets might subdivide the previous sentence level concept into such components as "There is a king", "The king's name is Claudius", "The country the king governs is Denmark", and "Claudius is

a character in a play". In these systems, the educator is required to construct a list of concepts which make up the answers to their question prompts, often this entails manually creating or generating many variations of the same concept to capture all possible student answers as well as requiring educators to have various degrees of linguistic proficiency.

Typical systems which utilize concept mapping require a lot of initial overhead and development by professionals in order to bootstrap viable solutions, for example, Burstein et al. (1996) build a concept-based lexicon and concept grammar before grading student responses [BJ99].

This brings up an important trade-off, namely, teacher time, input and expertise on the one hand and accuracy on the other. One can build more accurate ASAG systems with increases in detailed input, often at the cost of generalizability of the solutions. For instance, a teacher could simply specify sets of thousands of correct sentence fragments containing grammatical and semantic variations of correct responses which would increase the accuracy of the system but would come at the cost of proportionate increases in effort by the teacher.

The Era of Information Extraction

Following in the path laid out by the concept mapping phase, information extraction techniques require educators to create templates which hold empty slots for facts. ASAG is then viewed as a search problem whereby an algorithm attempts to find information in student responses which can fill the template slots the educator has defined. The methods employed are usually pattern matching operations which make use of regular expressions and parse trees [BGS14].

A prototypical example from this period is **Automark**. One of its templates is presented in Fig. 1.2. It is a system in which educators design templates consisting of a parse tree in which each node can hold one or more synonyms. Student responses are then parsed and matched against the templates to determine a score [TMA02].

One beneficial feature of these first two eras is that since they explicitly define concepts or templates and, although cumbersome and unruly to create, they can provide for excellent student feedback. Taking Fig. 1.2 as an example, a student who answers the question prompt with *"The earth moves in front of the sun"* could be given very detailed and pertinent feedback specific to his response. In this case, perhaps a note about the earth's orbital motion having a categorically elliptical quality.

Disadvantages of the approaches of both eras tend to be a brittleness in the face of variation and noise in student responses as well as complex and time consuming template and concept building tasks which also require a high degree of expertise and training on the part of educators.

The Era of Corpus Based Methods

> "You shall know a word by the company it keeps."
> — J.R. Firth, A synopsis of linguistic theory 1930-1955 (1957)

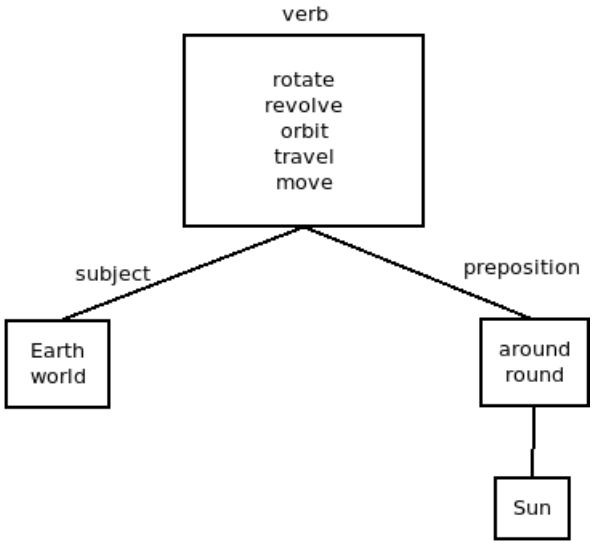

Fig. 1.2. Automark: an example of a template based information extraction system for ASAG. Shows possible values for the various parts of the template sentence: subject, verb and preposition.

Corpus based methods use statistical approaches to extract language properties determined by large collections of documents known as **corpora**. These methods are normally used for applications which have longer texts but many of them can be applied to short-text problems in ASAG [BGS14]. They are based around the idea that the meaning of a word can be found in the set of all the contexts in which that word occurs. A typical use case scenario would be to extend a vocabulary by discovering synonyms to words in the teacher created concepts or template databases by looking at many corpora and analyzing the frequency with which other words appear in the same contexts as a given concept word and then selecting the words which appear with the highest frequencies. Optionally, one can refine that selection afterwards.

One of the techniques which many of the systems of this era utilize is Latent Semantic Analysis (LSA) or a variant thereof, which is essentially a dimensionality reduction technique that starts by constructing a word by document matrix of word frequencies (rows) per document (columns) and uses single value decomposition (SVD) to reduce the number of rows i.e. words, thereby condensing the semantic space into underlying concepts which best identify the documents. In ASAG these documents are often defined as a collection of pregraded student responses and a given new student response can be compared using any vector space distance metric, typically the cosine distance, between itself and all other pregraded student responses in order to determine a score [MM09].

A key insight from this era is that corpus based techniques can improve short-answer grading models by incorporating deeper semantic relationships between words above and beyond simple lexical structure or pattern matching.

The Era of Machine Learning

Machine learning techniques in ASAG use natural language processing methods to extract sets of features from the input data - ordinarily educator created model answers or other precrafted question data but also from student responses themselves - and incorporate these features through a regression or classification model to produce a score as output for a given student short text response [BGS14]. Some common NLP methods which are used beyond various preprocessing techniques such as lemmatization, stemming and stopword removal are n-grams and bag-of-words methods. Typical machine learning algorithms are decision trees and support vector machines (SVM) which are developed using supervised or semi-supervised training methods.

The Era of Evaluation

One problem that plagues every previous era is the lack of standardized shared datasets for evaluating the results of disparate methodological approaches. The era of evaluation can be seen as a response to this problem and differs from the previous eras in that it is not marked by a particular methodology but by a creation, participation and dissemination of collections of datasets which are then utilized in various research endeavors and enable more fruitful cross-comparisons of scientific results [BGS14]. This promulgation of resources does not genuinely initiate until around 2012 when Kaggle, a commercial data science competitions company, hosted a competition for ASAP (Automated Student Assessment Prize). However, there are a few widely utilized datasets released prior to this date, such as the Mihalcea dataset released in full around 2011 [MBM11].

The methods applied during this period cover the entire spectrum of previous eras, combinations thereof, and even a few recent papers which have delved into the prospects of deep learning for ASAG such as using autoencoders to avoid the necessity to create target answers [YHZ+18].

Summary of the historical development of ASAG

Each phase of ASAG has carved up more precisely the problem space and shed light on various approaches to grading short-text responses. ASAG is clearly a hard problem when one considers it has been over 50 years since Page first outlined his arguments for automated essay grading in 1966 and although much progress has been made there remains quite a distance to travel in order to overcome all the complex issues the domain presents. These matters and fundamental obstacles with which ASAG has to cope, finding the balance between trade-offs and the specific focal points of this thesis as it pertains to ASAG constitute the material of the next section.

1.1.4 Key Problems and Issues

The main problem areas and difficulties within ASAG can be classified under the following five bullet points:

1. Text Input Correction and Normalization
2. Natural Language Understanding
3. Feedback Generation
4. Item Scoring
5. Language Independence

The final item, *Language Independence*, is not often discussed in the literature, since most ASAG systems have been developed for a single language, with the notable exception of the Comparing Meaning in Context project (CoMiC). It has two different systems each designed using the same techniques but whereas one system supports German language ASAG the other processes English language responses [MZOK11]. However, for this thesis, having a flexible and modular system capable of processing student responses in some number $n > 1$ natural languages is important for reasons later discussed.

Text Input Correction and Normalization

Below is an example real-world student response to a GCSE biology question, which demonstrates some of the issues involved in text correction and normalization.

> "The blood vessels stops a large ammount of blood going to the blood capillary and sweat gland. This prents the presonne from sweating and loosing heat".

In this example, there are grammatical mistakes such as the wrong grammatical number for the verb *stops* in addition to misspellings such as *ammount*, *prents* instead of *presents*, *presonne* for *person*, and *loosing* in place of *losing* [PS05]. These problems cannot be readily remedied since spelling and grammar correction is normally an interactive process in which suggestions are made and a human selects the appropriate correction. Furthermore, the number of word error rates in typed text has been reported to be between 1.5% and 2.5% [Kuk]. This means that from spelling mistakes alone, noise is introduced into the dataset, which an ASAG system should take into account. Another problem related to spelling correction is that automated spelling correction techniques use dictionaries for detecting non-word errors like *pinisilin* instead of *penicillin*, which are highly dependent on the existence of correct words in dictionaries whose word list vary by domain [Kuk]. For instance, a molecular biology ASAG system will need access to a much different spelling correction dictionary than a western history ASAG system.

Additionally, although spelling and grammatical mistakes are an issue, there are also numerous other input related problems which present themselves. For instance, consider the temporal expressions in various student responses containing the phrases "at 9am", "at nine a.m." and "at 09:00 hours" and a reference answer of "at nine o'clock in the morning". All of these are clearly referencing the same point in time yet resolving these entities, a process known as Named Entity Recognition (NER), is necessary in order to correctly grade the student response. Further, since

the tokens themselves are all different from one another i.e. “9am” is not the same token as “nine a.m.” some form of token transformation will need to be made in order to convert these tokens into a standard format, perhaps by normalizing every recognized entity to “at 09:00”. Other salient problems revolving around NER and ASAG are resolving not only temporal phrases but also such numbers and measuring units.

Besides these more exotic but none-the-less normalization and preprocessing steps, there are the more mundane processes such as tokenization, case normalization, stopword removal, stemming and lemmatization which are widely applied in NLP to normalize input data into a standard format. These techniques are basic to the task and although the particular parameter choices are important to model accuracy they don't present a solution to unique problems for ASAG and will consequently not be discussed in further detail here, though their details are discussed in Ch. 3.

Natural Language Understanding

Natural language understanding is the bottleneck for ASAG. Understanding text is a hard problem, especially when the contextual knowledge is limited like it is in ASAG [RND15]. Rather than looking broadly at all the problems in NLU, only those which are particularly notable in the context of student responses and reference answers are examined in this section, since the field is broad and thus necessitates a filtering for analytic purposes. First, there are the following problems that reach beyond state-of-the-art solutions and are identified by Pulman in 2005 [PS05].

1. **The need for reasoning and inference making**: For instance, given a model answer of *no fire*, a student may respond with *no wood or flint* which receives a failing grade unless there is some mechanism to infer that this means *no fire*.
2. **Students' usage of double negations for affirmative**: For instance, an answer such as *will not be done at any specific time* is equivalent to *can be done at any time*. Pulman's reasoning for giving this as a separate category is that the wording is very similar, however, in the general case as in the first example, the wording can be completely different.
3. **Contradictory or inconsistent information**: First, there are logical contradictions which could fool many simple ASAG systems built off of BoW techniques, such as *needs fertilization and does not need fertilization*, but additionally there are more complex contradictions which also need to be identified such as the phrase *identical twins have the same chromosomes but different DNA*.
4. **Unconventional expressions of scientific knowledge**: For instance, *the sperm and egg find each other* instead of *fertilization*.

These are almost exclusively knowledge, reasoning and inference related problems, broad in their scope, which push the envelope beyond the capabilities of current systems [PS05]. Stepping away from the context of reasoning and inference,

however, and into more constrained semantic domains allows for the inspection of NLU problems which are more approachable at the current state of technological advancement. They are:

1. **Word-to-Word Similarity**: consider the phrases m_1: *the earth rotates around the sun*, s_1: *the earth orbits around the sun*, and s_2: *the earth cranks around the sun*. The model answer m_1 and student answer s_1 are fully equivalent because *orbits* in this context has the same meaning as *rotates*, however, s_2 has a different meaning even though *cranks* is very closely related to *rotates*. This example illustrates a need for two capabilities, first, an understanding of synonymy i.e. an ASAG system should be able to look at a word like *orbits* and understand that it means roughly the same as *rotates*, at the most basic level, independently of the context in which the word occurs. Secondly, an more advanced system would also take context of the surrounding words into account, since in many cases *cranks* will suffice as a valid substitute, for example, in the phrases *it cranks the wheel* and *it rotates the wheel*.
2. **Syntacto-Semantic Meaning**: often the syntactic structure of a phrase can heavily influence the meaning of that phrase. For example, although the phrases *The sperm finds the egg* and *The egg finds the sperm* contain the exact same words they have completely different meanings. This difference in meaning arises from word order or the syntax of the two phrases. Thus, an ASAG system needs to take syntactic structure into account if it wants to correctly understand meaning.
3. **Anaphora Resolution**: continuing with the example of the earth's orbit, consider the phrases *it rotates around the sun* or *The third planet, it rotates around the sun*. Any human will immediately recognize that *it* in both phrases refers to the *earth* even though in the second sentence *it* is referencing *the third planet* which is itself a synonym for the earth. Resolving these pronouns to understand how the various placeholders relate to one another is a complicated but approachable issue in natural language understanding. However, it is unclear how relevant resolving these references is to the task of ASAG.

Feedback Generation

Feedback generation, in the author's opinion, is an underresesarched domain within the literature on ASAG, although in its sibling field "Automated Essay Grading" much has been published on diagnostic feedback [YA06]. What is the use of a numeric value applied to a test item if it is not transparent to both teachers and students as to how that score is derived? Further, a score should provide a justification based on evidence for its existence. If an item is scored with a 7 on a scale from $1 - 10$, it should give evidence for how it arrived at that particular score instead of a 0 or a 10. Much of the purpose of grading is to provide formative feedback to students so that they may improve their understanding of the material and performance in the given domain and to inculcate the subject matter in more effective and sophisticated ways in students' minds. Granted there is efficiency to

be gained by building an ASAG system that imitates the behavior of a teacher even without detailed feedback or intuitive scoring diagnostics, but it is difficult to imagine that more transparency in the scoring mechanisms could be anything but positive. For instance, even MOOCs where a teacher may not be available at all, still an administrator or the students themselves want to know why they are receiving a particular score.

There is apt reason, however, that feedback is largely ignored in the literature. In the author's view, it is because feedback requires many of the reasoning and inferential skills mentioned above which are beyond the state-of-the-art, but also because it is very difficult to quantify qualitative feedback and compare results across researchers. Nonetheless, there have been various Intelligent Tutoring Systems (ITS) developed which utilize one or more of the following techniques.

1. **Hinting**: is the situation in which a student enters an incorrect answer and afterwards receives a teacher constructed hint in the form of a text or image. Then the student has the opportunity to attempt the question again for an often consequently reduced point score. This approach is characteristic of many real-world online learning platforms such as Khan Academy and DataCamp [KA18, Dat18].
2. **Correct Response**: simply refers to showing the correct answer after the student has missed the question. The idea is that by seeing the correct answer the student will be able to understand why his submission was incorrect and improve on future question items.
3. **Dialogue**: some ITS have a dialogue with students to assess and help them understand concepts, much like a one-on-one interaction with a teacher. If a student gets an answer incorrect these systems attempt a natural language dialogue to aid the student towards understanding. A classic example of this approach is AutoTutor which attempts to simulate a human tutor by having a natural language conversation with a learner [GCHO05]. To the author's knowledge, this technique has never been used in an Online LMS, especially for short-answer responses, however, it is a method which, although difficult to implement, at first glance seems beneficial from the students' perspective.

Item Scoring

If all the normalization, language understanding and feedback mechanisms have been dealt with, the main lingering issue is item scoring of student responses. Many methods have been employed in the literature and these can be broadly broken down into two categories: *Rule-Based Methods* and *Statistical Methods*, the former being found in the eras of Concept Mapping and Information Retrieval and the later statistical approach arising in the eras of Corpus Based Models and Machine Learning [BGS14]. Each of these methods has particular salient properties making it more applicable for certain assessment types than others. Rule-based methods boil down to specific rules on a question by question basis and can be better suited for repeated assessments because although the cost of initial setup is higher and the

grading component is less generalizable, these costs come with the benefit of high accuracies for grading that particular question item [BGS14]. Conversely, statistical approaches are more generalizable and flourish when the grading task involves previously unseen questions and domains making this approach more appealing in the context of MOOCs which deal with a diverse assortment of questions and noise filled answers [BGS14].

Finally, there is the relationship between scoring and feedback. In the author's opinion, feedback should be tightly integrated with scoring because, as has been discussed, scoring without feedback is neither formative nor is the summative information it provides very useful.

Language Independence

This is not a concern for any of the ASAG systems which have been encountered in the literature or in perusals of internet resources by the author. However, there could be other systems which have simply not been discovered that also have to deal with this issue. The problem originates when an LMS supporting ASAG needs to be capable of being used in multiple languages. The ASAG systems which have been encountered are usually either one-off research attempts in a specific language such as English or German or they are open market systems developed by a corporation such as ETS which only provides services in a single language.

Stembord, the LMS in which the ASAG module is integrated, is unique in that it is designed from the ground up to be capable of supporting multiple languages. In its current incarnation it supports only English. However, Spanish as well as German language support is planned and, therefore, ideally any ASAG solution should be capable of easily incorporating new languages as support for them is added to the platform. This software engineering concern does eliminate the ability to use certain software libraries which are only available for one language, usually English, as well as forcing the ASAG module to have a design which maximizes the amount of shared functionality across languages and minimizes the degree to which each language needs language specific solutions.

1.1.5 Current State-of-the-Art

Having seen the array of complications and hurdles imposed by natural language phenomena in the field of ASAG and the noise encountered in short-answer responses, now the question arises as to the peak of ASAGs capabilities, essentially, the question is posed: what is the current state-of-the-art for ASAG systems?

This is a tough question to answer, since most ASAG systems to date have been evaluated in isolation with few datasets in common between them [BGS14]. However, in their detailed analysis, Burrows et al. conclude that concept mapping and information extraction techniques outperform other machine learning techniques and have higher precision when the assessment design is not required to perform holistic marking, however, this is at the cost of effort in creation time [BGS14]. They further conclude that for statistical machine learning approaches the Mohler

'09 system performs best, particularly on unseen questions and responses and that there is a trade-off in the effectiveness of models between repeated/non-repeated assessment and seen/unseen questions and domains with IE and concept mapping performing best for repeated assessment and seen domains and machine learning for unseen and non-repeated domains [BGS14].

1.2 Learning Management Systems (LMS)

This section forays into the depths of computational learning systems. Over that journey an exploration of concepts in Learning Management Systems (LMS), Intelligent Tutoring Systems (ITS) and the role of artificial intelligence in education is conducted in order to elucidate the functionality of Stembord, an Intelligent Learning System, to argue for the necessity of the chosen methods and to characterize the context imposed by the Stembord system on current and future development of an ASAG module.

1.2.1 Definition and Function of LMS Software

In the 1950s, long before computers began to clutter the pockets and desks of society, computational technology was already being applied in education [WRW07]. A Learning Management System (LMS) is one approach of many towards the application of computers in the educational domain. Aside from LMSs, there are many other generic terms in the literature for this circumstance, such as computer-based instruction (CBI), computer-assisted instruction (CAI) and computer-assisted learning (CAL), all of which outline various programs for drilling exercises or individualizing instruction [WRW07]. The distinction between these terms and LMS is that an LMS goes beyond instructional content and offers management, tracking, personalized instruction and integration across the entire system [WRW07]. A LMS is a framework that handles every facet of the learning process in an organization or institution. It delivers the instructional content, assesses and identifies individual goals and training. LMSs track individual and classroom progress and collect and present interpretable data so that students and educators, through clearer information, can better understand the various states of knowledge and skill acquisition by users of the system [WRW07].

There is much confusion in the literature as to the precise nature of an LMS and its distinguishing characteristics from other similar technologies such as Content Management Systems (CMS), for example Blackboard, and Learning Content Management Systems (LCMS). The key difference is that an LMS manages the learning process as a whole, it provides the rules, whereas a CMS and LCMS are devoted to the content and learning objects (LO) themselves such as creating lessons and tutorials - they provide the content [WRW07].

Some of the key features which most LMS solutions provide are listed below [Cjb05].

1. Asynchronous and synchronous communication (chat, e-email, instant messaging, discussion forums)
2. Content development and delivery (Learning Objects, links to internet resources)
3. Formative and summative assessment (assignment submission, testing, collaborative work and feedback)
4. Class and user management (registration, enrollment, timetables, managing student activities)

These features demonstrate the broad reaching scope of LMSs as these systems are designed to support the productivity and effectiveness of all stakeholders in the educational context: students, teachers and administrators alike.

Learning Objects

Learning objects (LO) are the smallest unit of content within a CMS or LCMS. They are powerful because they offer reusability across systems and contexts [WRW07]. An example of a learning object could be a lesson on "Taking square roots" or a quiz on "The biological organization of the cell". Each of these LOs can be used within a classroom, but also shared and used by other teachers and educators within their classrooms, or even be studied solo by a student simply interested in learning about taking the square roots of numbers or understanding the structural organization of a cell. Overall, there are four properties of LOs. They are:

1. **Reusability**: a LO once created can be used in any course forever.
2. **Generativity**: a LO or quiz can be used to create new instructional content.
3. **Adaptability**: a LO can be given to a student based on his skill level.
4. **Scalability**: a LO can scale to meet the needs of small or large audiences without an increase in costs.

It is important to note that LOs are not just lessons but can be quizzes, interactive visualizations and even full exams, that is, they are modular structural units of teaching and learning. They are the building blocks, the grammar which provides the combinatorial productions of the language of learning systems.

ASAG's relationship and role in LMSs

In the LMS, LCMS or CMS systems on the market, short-answer responses are, to the best of the author's knowledge, ubiquitously introduced as a further question type which teachers have for designing their LO quizzes and exams. Curricula designers create a question of type *short-answer* and then have an options menu for configuring the question. Some systems such as Moodle allow for automatic grading of the student responses. In Moodle, for instance, the teacher can create a list of regular expressions which are then matched against student input to calculate a score [Moo18]. Blackboard, the most widely used CMS system for education in the US, as well as other open source solutions such as OpenOLAT do not provide ASAG, however, Blackboard still requires teachers to enter a reference answer in order to give students formative feedback [Bla18, WRW07]. From a software design perspective, most of these production systems are closed source which makes it impossible to assess how they incorporate a short-answer module into the architectural design of the fully realized system, however, Moodle is a notable open-source exception which includes a *short-answer* module.

Reasons for LMS usage and adoption uptake

Learning Management Systems have seen rapid adoption across the globe despite the complexities, costs and risks involved in the adoption process [Cjb05]. When an institution endeavors to adopt a LMS, it is not partaking in a low-risk decision making process. This resolution involves institutional and technological forecasting, restructuring of complicated administrative, educational and technological issues, a consideration of the interests of a deluge of disparate stakeholders and often a reconsideration of institutional policies and procedures, not to mention new guidelines for accountability and control [Cjb05].

Despite all of this, many universities have decided to adopt LMS solutions. Clearly, these institutions see potential for increased productivity along some dimensions otherwise they would not take the risks and undergo the hurdles in order to adopt these systems. Broadly speaking, the most commonly listed reasons for adoption are access, cost and quality although these core areas can be subdivided into a plethora of more specific factors [Cjb05].

A detailed assessment of the varied reasons for LMS adoption is tangential to the subject of this thesis, however, to list just a few, Coates identifies such factors as a means of delivering large-scale resource-based learning programs, flexible course delivery, increased teaching efficiency, reduction in course management overheads, enriched student learning, automatic and adaptive assessment and competitive pressure between institutions to meet demands [Cjb05].

Regardless of the causes, LMSs have permeated the educational domain and it appears they will stay for a while, transforming and interacting with the academic landscape into the future.

1.2.2 Definition and Function of Intelligent Tutoring Systems

The goals of Intelligent Tutoring Systems (ITS) are similar to those of an LMS: to provide services which support learning, however, for an ITS these services are specifically tailored for the tutoring context [NMB13]. Research in the field began in the 1960s and '70s with the term "Intelligent Tutoring Systems" eventually being coined by Sleeman and Brown in 1982, followed in the same decade by the first ITS conference in 1988. One motivator for the field was research published by Bloom in 1984 which demonstrated that individual tutoring is twice as effective as group teaching [Blo84, NMB13]. In the first two decades of its incipient rise, ITS was further driven by visionaries who dreamed that every child would "have access to ... the personal services of a tutor as well informed as Aristotle" (Suppes, quoted in Nwana 1990). Into the 90s ITS had begun to establish itself as an engineering design field with scientific foundations to its research [NMB13].

In order to model tutoring, ITS use a four-component architecture which can be seen in Fig. 1.3. The fourth component is the user interface and is ignored in this section since its not useful in understanding how intelligent systems function as a whole.

The **domain model** contains the knowledge base and reasoning faculties. It stores the concepts and rules as well as problem-solving strategies within the do-

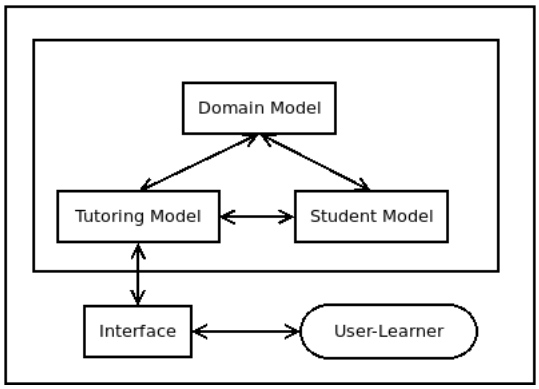

Fig. 1.3. The four-component architecture view of ITS.

main to be learned and is expected to adapt explanations of its reasoning to the learner. Knowledge elements in the domain model can be linked together as lessons or more loosely as dynamic curricula. There are many different structures for storing this information such as semantic networks, frames, ontologies and production rules [NMB13].

The **student model** stores knowledge about the student's affective state, his current knowledge on the topic and his evolution through the learning process. The student model must gather explicit and implicit data about the learner, synthesize that data to create representations of the student's knowledge and state in the learning process, and it must account for the data by performing diagnosis and selecting optimal pedagogical strategies for the presentation of subsequent learning materials to the student [NMB13].

Lastly, the **tutoring model** interacts with the student and domain models and determines which actions and tutoring strategies to take. It must make decisions about if and when to intervene, and how and when it should deliver learning content. These can be Socratic dialogues, feedback or hints, but also simulations and visualizations [NMB13].

1.2.3 Stembord Project

Stembord is a combination of an LMS and an ITS. Its goal is to synergize the concepts of ITS and LMS into a system which provides learners and educators with resources optimized for their educational needs and adapted to their cognitive states.[1]

Immediate and Long-Term goals of Stembord

Some of the long-term goals of Stembord from a single student's perspective entail such features as adaptive and dynamic lessons, student progress and affective modeling, knowledge construction, natural language dialogue in multiple languages

[1] A short piece of prose outlining the Stembord project's vision for the future can be found in the appendix in section A.4.1

and high availability for students. However, students are not the only stakeholders Stembord is concerned with. Others are parents, teachers, as well as school administrators, each of which would benefit enormously from systems designed with the tenets of cognitive psychology to promote transparency, engagement, mental growth and reduce bias. However, in order to achieve these long-term goals, many diverse building blocks are necessary. Chief among these is a need for quality LOs such as quizzes, guides and interactive simulations in every domain from physics to literature, not to mention labeled knowledge bases and datasets for handling one-on-one student learning sessions, eventually between a Stembord AI tutor and the student. Yet before any query can be made by an AI tutor, there must also exist a database over which to query. Stembord also needs to build student models so that it can discern the affective and progress states of each student, querying and adapting lesson plans and learning pathways to the skill-sets, faculties and needs of each student dynamically.

In order to achieve these far reaching goals, Stembord has decided to focus initially on the educator side of the problem by building tools which allow teachers to design eloquent and semantically labeled LOs and combine them to construct online courses, exams and guided tutorials. From an ITS view, one could see this as a focus on building a broad and deep domain model backed by a robust knowledge base first. Only later, is the goal to then progressively add student modeling, interactive natural language dialogue, the tutor model, as well as school administration capabilities such as enrollment, scheduling, etc. None of these later features are currently implemented in Stembord. However, any new module, particularly the ASAG module, must take these long-term architectural goals into account in order to be an effective component in future versions of the end-product.

In the near future then, Stembord is focused on teachers and their needs in the classroom. One of these needs is quiz and test creation and management functionality. This is where ASAG comes into play. With the current quiz-maker in Stembord, teachers can create multiple-choice and true-false questions and have them automatically graded, yet this capability does not currently exist for short-answer response question types. Therefore, this thesis researches the problem of ASAG and of integration of an ASAG module into this quiz-maker system of Stembord.

Also of importance, is that Stembord should eventually offer all its functionality in as many of the world's languages as possible. The reasoning for this is simple, namely, that the developers of Stembord want anyone regardless of location or native language to be able to acquire knowledge they need to improve their lives with as little friction and as low a cost as possible. Providing Stembord resources, features and functionality which works for a user's native language serves to reduce barriers and lower hurdles which are irrelevant to the user's main task which is acquiring knowledge about some topic, for instance, mathematics or computer programming.

2

Motivation and Problem Definition

> "I learned very early the difference between knowing the name of something and knowing something."
>
> — Richard Feynman

Up until this point important background information about Automatic Short Answer Grading, Learning Management Systems and particularly Stembord has been expounded upon in order to provide a clear backdrop for understanding the problem definition and key motivations presented in this chapter. Elucidating and clarifying the problem of ASAG, informing and developing the problem definition and, additionally, expounding the motivation for tackling this problem are the main subjects of this chapter.

2.1 Problem Definition

Short-answer question items are the core components of ASAG and consist of three key parts: a question prompt, one or more model answers written by educators and a student response. These are visually displayed in Fig. 2.1.

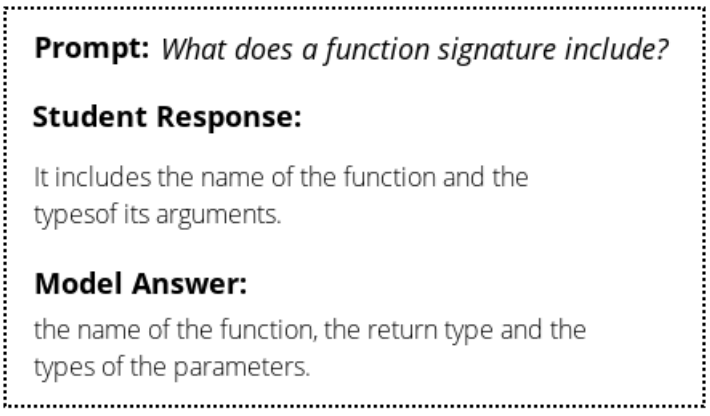

Fig. 2.1. Example showing the three core components of short-answer question items: The prompt, teacher created model answer and student response.

More formally, for the purposes of this thesis, the problem of Automatic Short Answer Grading is defined as follows:

Definition 2.1. *Given a set M of reference short text responses such that $|M| \geq 1$ and a short-text student response S, develop a function $f(M, S)$ such that $0 \leq f(M, S) \leq 1$. Where $f(M, S) = 1$ means S is semantically identical to M and $f(M, S) = 0$ means S is completely unrelated to M. Values ranging between 0 and 1 represent a gradient of increasing shared meaning between S and M as $f(M, S)$ approaches 1.*

Developing the function $f(M, S)$ is the main goal of this thesis. Concrete examples of the three components given differing student responses and consequent scores are shown in Fig. 2.2.

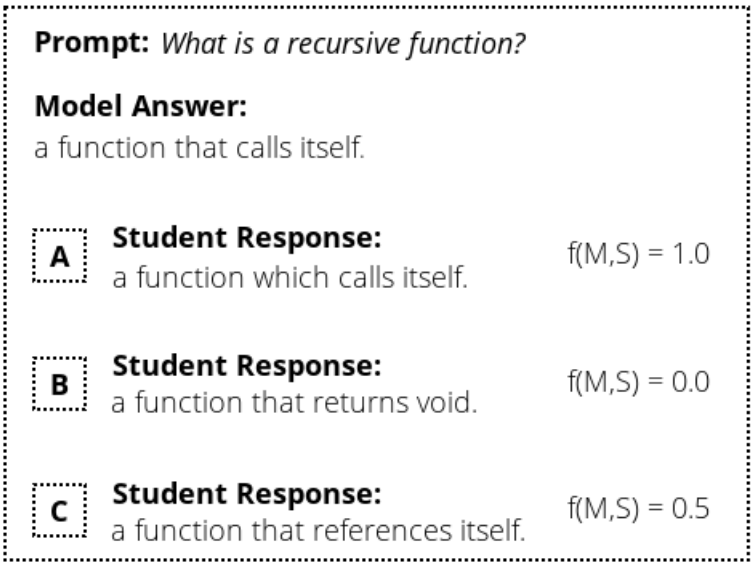

Fig. 2.2. Demostration of how the function $f(M, S)$ should work. The three possible scoring situations for student responses are shown. **A** shows a correct student response which consequently receives a perfect score (1.0). **B** shows a false answer which gets a score of 0.0. And **C** shows a student response that is only partially correct. It consequently receives a score of 0.5.

The reason the range of the function $f(M, S)$ is $[0, 1]$, is that any output of such a function can easily be scaled to any grade value. For instance, if a teacher assigns 20 points to a short-answer question and $f(M, S)$ calculates a similarity score of 0.7 between model-answer M and student-response S then the student can straightforwardly receive 14 points ($0.7 \cdot 20 = 14$). If the original point total were 10 or 5, the score would remain just as easy to calculate, except that for 5 some rounding would likely need to take place if the desired output score is an integer value.

There are many other ways to define the problem of assigning a score to a short text response submitted by a student. Indeed, in some of the experiments in this thesis the set M is modified to be, for instance, a set of patterns written by teachers. Nonetheless, the structure of the problem stays the same and the range of its output on a continuous scale remains in essence equivalent. This is not to say this is the only correct approach or that other comparably effective approaches are not possible. This definition principally serves merely as a formal grounding for understanding the problem and the main task undertaken in this thesis.

2.2 Motivation

Why is developing such a function $f(M, S)$ useful and what holes in the current knowledge landscape does such research help to fill? Some of the answers to this question have been variously hinted upon in previous sections, however, the purpose of this section is to concretize those claims. Broadly speaking, the motivation for this research comes from two areas. The first, is the field of ASAG itself in which many open questions still remain, and the second comes from Stembord which in some ways sheds new light on old problems and unearths novel motivational questions and constraints upon the research endeavour.

2.2.1 Motivation from ASAG

Much research on ASAG has hitherto been conducted analyzing specific techniques, whether supervised or unsupervised, with the explicit goal of improving correlations to human graders. This means there is a lot of work on technical and statistical approaches to automatic grading of short-text responses, and this is certainly a positive attribute of the field. However, the author has not been able to find detailed analyses on the structure and topology of short-text responses in the context of automatic grading and even less on the relationships between *(model-answer, student-response)* pairs. There could be many reasons for this, first and foremost, inadequacy and failures on the part of the author in his process of literary research. Another possibility is that these structural analyses have been conducted but either there is not much information which can be gleened from them or such information is so rudimentary that it is implicitly assumed in most published papers and ASAG researchers simply know their collegues already have this basic topological knowledge and write with an implicit assumption of this basis.

Whatever the causes, one reason for carrying out this research is to answer the following questions:

1. What are typical structures that emerge in short-text responses during Short Answer Grading, particulary those relevant for automation of the grading process?
2. How do structures within short-text responses influence a human grader's behaviour?

3. Are their systematic relationships between model answers and student responses and if so what are they?
4. Are their systematic mistakes students tend to make when responding and if so what are they?
5. Can these structures, relationships, mistakes and grading behaviors be ordered in structured taxonomies to provide additional insight?

Knowing the answers to, at least, some of these questions could provide useful insights for further research as well as new ideas for current technical and statistical approaches to ASAG, not to mention serve as a helpful resource for discussing common short-answer response patterns and behaviours.

But this is certainly not the end. Most research until now has focused either on techniques or on isolated systems with little if any afterthought about how the ASAG solution can be integrated into a full LMS or learning context. This is not a criticism of the current research, indeed, isolating problems and testing specific hypotheses in a controlled and manageable setting is paramount to making progress. It is only mentioned here, because it is a unique perspective offered by this thesis, since it looks at ASAG from a more holistic vantage point by building a module and integrating that module into the Stembord LMS. This holistic view provides some additional motivations and angles from which to inspect the problem.

2.2.2 Motivation from Stembord

In the introductory chapter, the needs of teachers in the context of Stembord's current architecture were explained. Chief among these is that the effort required to construct learning objects be kept to a minimum. Therefore, an informed research should focus on discovering effective ASAG techniques which are user friendly and do not require large amounts of technical expertise or laborious and time consuming efforts on the part of teachers. Most approaches which meet these requirements are largely unsupervised in nature. Therefore, the majority of this thesis focuses on unsupervised short answer grading techniques and could be viewed as attempting to answer the question: what trade-offs exist among various unsupervised ASAG techniques?

Also, since Stembord coexist in a market shared by many other LMSs, it is important to have some form of a baseline for assessing the effectiveness of a standard approach to offering ASAG within an LMS, insofar as such an approach exists.:

Finally, because of the multilingual goals of Stembord, a major question is which techniques are applicable and effective in multiple languages and to what degree. Answering this question is important for an ASAG module being developed in the context of a multilingual software application but often irrelevant in the research literature. Even so, it may provide for unique insights.

These motivations, contributed by and arising from Stembord, are summarized below:

1. What are trade-offs among (unsupervised) ASAG techniques?

2. What is the effectiveness of a standard ASAG solution within another LMS on the market?
3. How do ASAG methods compare when it is necessary that they function across multiple languages?

2.2.3 Definition Specific Motivations

Finally, of the main research questions mentioned above, subproblems within the version of ASAG given in this chapter's definition proffer up their own motivational questions, the most pertinent of which are:

1. How does increasing the number of model answers M affect the ability of an ASAG solution?
2. What is the best ASAG method for extracting meaning from the words of a sentence?
3. How do knowledge based measures of word meaning using *WordNet* compare to word embedding based measures and what are the key differences and trade-offs?
4. What is the best unsupervised i.e. algorithmic approach to ASAG that does not rely on machine learning?
5. What machine learning model is best for supervised ASAG using discoveries made in this thesis?

This represents a comprehensive list of core questions which are explored in the coming pages in this thesis, however, they are all derived from the main guiding force which posited is: how can one design a system to automatically grade short-text responses, build this module in software and integrate it into Stembord?

3

Material and Method

> "If we knew what it was we were doing, it would not be called research would it?"
>
> — Albert Einstein

This chapter presents the data, its annotations and extensions and insights gained from explorative data analysis. It lays out performance boundaries for expected good and bad automatic grading models. It describes, in cursory detail, standard natual language processing methods utilized and, in more extensive detail, methods developed and applied in the experiments. It concludes by defining statistical and technical measurements needed to evaluate the results of the experiments.

3.1 Datasets

3.1.1 Mohler Dataset Version 1.0

The Mohler dataset is used as the basis for running all the experiments. It was released publicly over two years in two separate versions and publications. The first version, version 1.0, is only used in the first experiment. It is the public dataset which Mohler and Mihalcea published along with their paper, *"Text-to-text Semantic Similarity for Automatic Short Answer Grading"* [MM09]. Version 1.0 (v1.0) contains 21 short-answer question prompts of which 17 are also present in version 2.0 (v2.0). The reason v1.0 was chosen over 2.0 for the first experiment is that the former only contains 21 unique model answers and the experiment design calls for a high degree of teacher effort creating model answers, therefore, the lower question count of 21 provides lower time requirements for experiment participants.

Both datasets contain questions from an introductory computer science course for undergraduate students offered at the University of North Texas. Assignments were taken by students using the WebCT online learning environment and the student responses to the short-answer questions were collected via this online tool. For version 1.0, there were 30 students enrolled in the class who also took each assignment, 3 assignments in total were given with 7 questions per assignment for a grand total of $(30 \times 3 \times 7 = 630)$ student responses [MM09].

The student responses were graded independently by two human judges on a scale from 0 (false or incorrect) to 5 (completely correct). Both human judges were graduate students in the computer science department. These two scores were then averaged to create the final score for each student response item in the dataset. The inter-annotator correlation on the data set was .6443 using Pearson's correlation coefficient [MM09].

The dataset is publicy available in raw text format.[1] This raw text data was downloaded, parsed and converted into a comma-separated values (CSV) file containing the structure shown in Tab. 3.1. *AID* is a reference to the assignment ID, *QID* is the ID of the question number within the assignment, *SID* is the anonymized student ID, *Prompt* is the textual prompt for the question, *Model Answer* is the reference answer written by the teacher or educator, *Student Answer* is the student's response to the question item and *Score* is the averaged grade assigned by the two graduate student graders for that student's response.

The histogram and ECDF distribution of the student scores can be seen in Fig. 3.1, which visually displays the distribution of scores across v1.0 of the dataset.

This dataset is skewed in favor of high scores, in fact, any model which simply predicted a perfect score of 5 would be correct 45.4% of the time. This aspect of the dataset can be seen more distinctly in the ECDF (Empirical Cumulative Distribution Function) shown in the right of Fig. 3.1, which demonstrates that less than 40% of the responses make up the scores from 0 to 4, another way of conceptualizing this is that of 11 possible grades (0-5 inclusive at steps of 0.5),

[1] At the time of this writing, the raw datasets can be downloaded Dr. Mihalcea's website at `https://web.eecs.umich.edu/~mihalcea/downloads.html#saga`

AID	QID	SID	Prompt	Model Answer	Student Answer	Score
1	1	6	What does a function...	The name of the func...	It includes the name of the prog...	4.5
1	1	5	What does a function...	The name of the func...	It has specific information abou...	3.0
1	1	8	What does a function...	The name of the func...	The function signature includes ...	5.0
1	1	3	What does a function...	The name of the func...	A function signature consists of...	4.5
1	1	4	What does a function...	The name of the func...	It includes the name of the func...	5.0

Table 3.1. Structure of the CSV File for version 1.0 of the Mohler Dataset. Shows the columns: **AID** - assignment ID, **QID** - question ID, **SID** - student ID, **Prompt** - question prompt, **Model Answer** - the gold standard answer written by the educator, **Student Response** - the student's answer to the question and **Score** - the averaged score of two human graders.

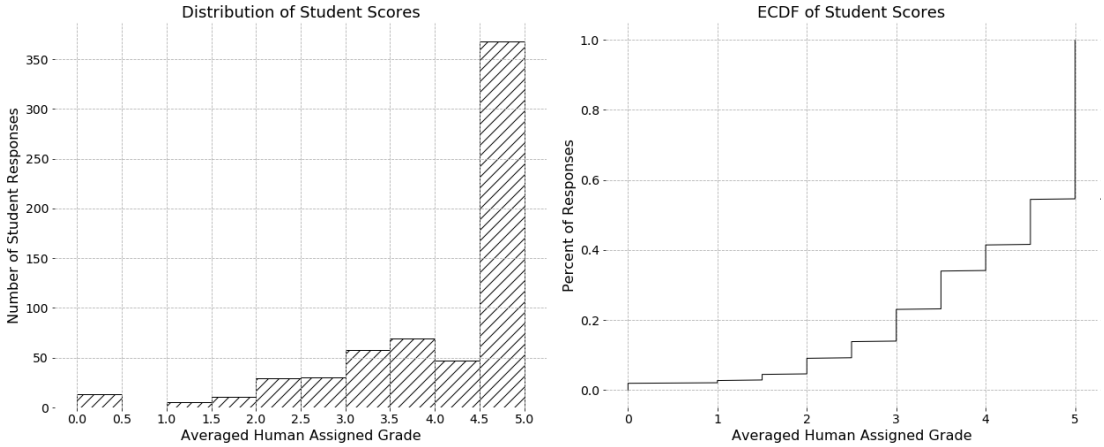

Fig. 3.1. The histogram and ECDF distribution of averaged student scores across v1.0 of the dataset.

3 grades (4, 4.5, and 5) make up over 60% of the dataset's responses, heavily skewing the dataset towards higher grades. However, this aspect of the data is also demonstrated in version 2.0 of the Mohler dataset and, as the author of this paper proposes, can be explained to some extent by the manner in which graders approach the short-answer grading task.

However, the skewed distribution provides an additional reason to emphasize the development of a system which stores student responses and builds a quality dataset going forward. It is also one of the reasons specific examples from each experiment are examined. This is done in order to elucidate problems or features and to develop intuition behind a method which may not be immediately visible by simply comparing correlation coefficients, MSEs or various other statistical and technical measurements.

Finally, because Stembord emphasizes self-study courses for its users, it is important to evaluate methods for ASAG which either pass or fail a student response.

This is because self-study courses do not have teachers and only need to detect whether a student has entered a correct response or not and provide the student with some degree of feedback. To build this dataset, all scores below a threshold of 2.5 where assigned the value of 0 for failure and scores such that $score \geq 2.5$ where assigned a value of 1.0 for success or passing. The threshold value of 2.5 was decided upon based on a detailed analysis of the underlying datasets, especially the augmented Mohler version 2.0 dataset.

Fig. 3.2 shows the histogram of the dataset after converting the scores to the binary classes.

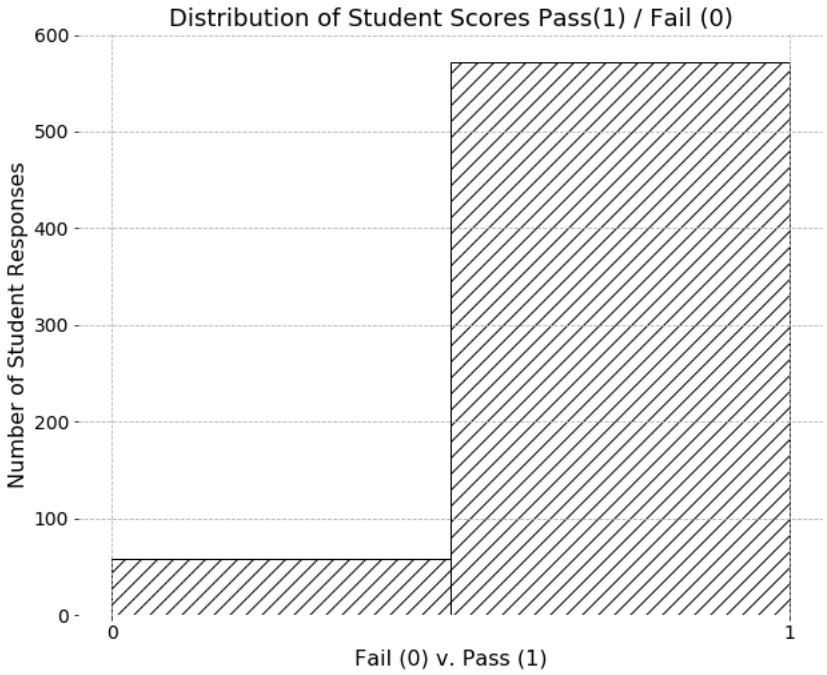

Fig. 3.2. The histogram distribution of the fail (0) and pass (1) scores across v1.0 of the dataset.

This concludes the description of the first version of the dataset. The key takeaways are that the scores are skewed to the left, with most student responses receiving a passing and high grade and that a threshold of 2.5 is used to divide human grader scores into failing and passing grades. The reasoning for these choices is discussed in the next section along with a detailed analyses of v2.0 of the dataset.

3.1.2 Mohler Dataset Version 2.0

In 2011, in their paper *"Learning to Grade Short Answer Questions using Semantic Similarity Measures and Dependency Graph Alignments"*, Mohler et al. released a second extended version of the original dataset. [MBM11]. This dataset expands

the student submitted responses to cover 80 questions across ten assignments and two examinations in a course on data structures at the same university. Although 31 students were enrolled in the class, some did not answer all the question items which results in the final dataset containing only 2442 items instead of the expected $80 \times 31 = 2480$. Just as in the original dataset, version 2.0 uses the average scores given by two human judges, both teaching assistants, to define the gold standard labels [MBM11].

Although this dataset is almost four times larger than version 1.0, it still exemplifies the same left skewedness over the distribution of the scores. These distributional properties of the raw dataset are illustrated in Fig. 3.3.

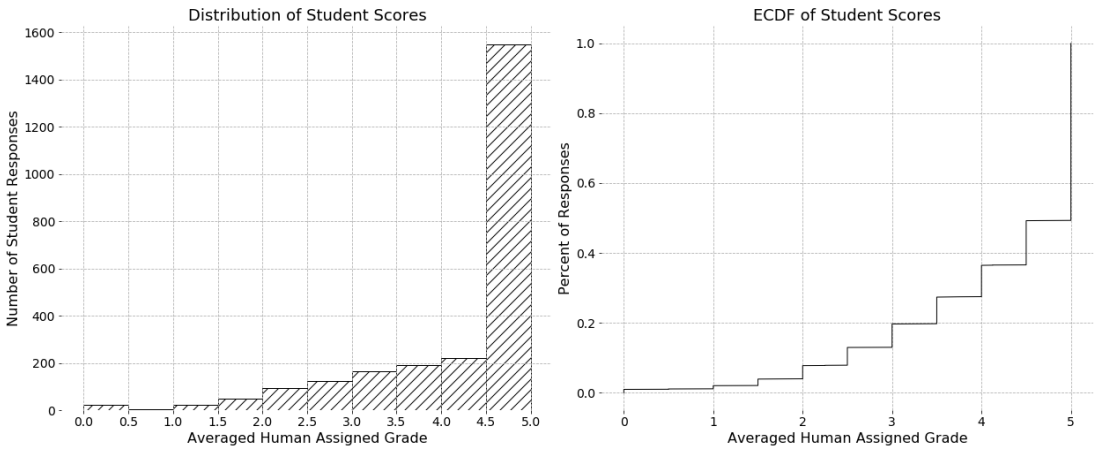

Fig. 3.3. The distribution of averaged student scores across v2.0 of the Mohler dataset shown by the Histogram and ECDF respectively.

Besides having more questions and response pairs, version 2.0 of the Mohler dataset is, structurally speaking, the same as version 1.0, that is, it has the same columns and attributes.

3.1.3 Annotation and Extension of Mohler Dataset Version 2.0

The extended, adapted and annotated version of the second Mohler dataset is used in all the experiments except those which use Mohler v1.0 and for this reason a detailed analysis of its structure as well as the corrective measures which were undertaken in order to ensure its robustness are disclosed in the following paragraphs.

Discoveries made by Hand-Grading the Dataset from Scratch

First, in order to find answers to motivational questions about emergent structures in short-text responses and how these structures can influence human grading behavior, the author hand-graded all of the 2442 student responses to gain an intuitive understanding of the grading process. The coefficient of these scores is

0.74, that is, the author's grades correlate 0.74 with the averaged scores of the two other human graders. After completing the original grading process, the author randomly selected $n = 100$ questions and graded them a second time in order to see how well the author's second attempt at grading correlated with his first one; this value was 0.80. Thus, in this small $n = 1$ task, a disagreement on assigned scores was shown to exist between the author at time t_1 and the same individual at a later time t_2. These results are consistent with some of the degrading arguments put forth by Kohn and discussed in the opening chapter [Koh99].

Grader Behavior and Student Response Classes

Overall, the process of grading proved to be difficult and time consuming. However, some parts of the process were easy. For instance, student responses which were highly similar to the model answer and correct were easy to assign a consistently high grade. Also those responses which were completely unrelated to the topic or where the student had not even written a response were easy to assign a 0 or bad score. These situations can be thought of as representing low effort extremes on the *(model-answer, student-response)* relationship landscape and these edge range decisions were simple. However, as soon as student answers began to deviate from the two patterns, difficulties emerged. One of the hardest problems to arise, was determining a consistent numeric value to give to a student's short-text answer, since in many cases student responses were quite similar to one another, yet minor differences caused significantly different grades. This led the author to quickly develop a classification system, whereby student responses could delegated to classes and then consistently graded according to their classification.

The classes were assigned labels which were derived over the process of the grading experience itself and were called: *wrong_unrelated*, *wrong_related*, *partially_correct*, *correct_indirect* and *correct_complete*. A description of each of these labels can be found in Tab. 3.2. The main goal behind this classification effort was to improve the author's grading consistency.

Label	**Description**
wrong_unrelated (WU)	Student either has no response or his response does not relate at all to the question.
wrong_related (WR)	Student's response is on the correct topic just completely wrong.
partially_correct (PC)	The response is on topic but not fully correct for any number of reasons.
correct_indirect (CI)	Response is fully correct but requires significant inference and external knowledge to infer this fact.
correct_complete (CC)	Response is fully correct, well worded, and easily interpretable.

Table 3.2. Custom labels applied to student responses in order to differentiate them into classes and aid in consistent scoring.

Once the entire dataset of student responses was assigned these class labels, the averaged score distributions from the human graders as well as the author when grouped by the classes could be cross-compared to see what patterns emerged. Fig. 3.4 shows the compared results of this grouping.

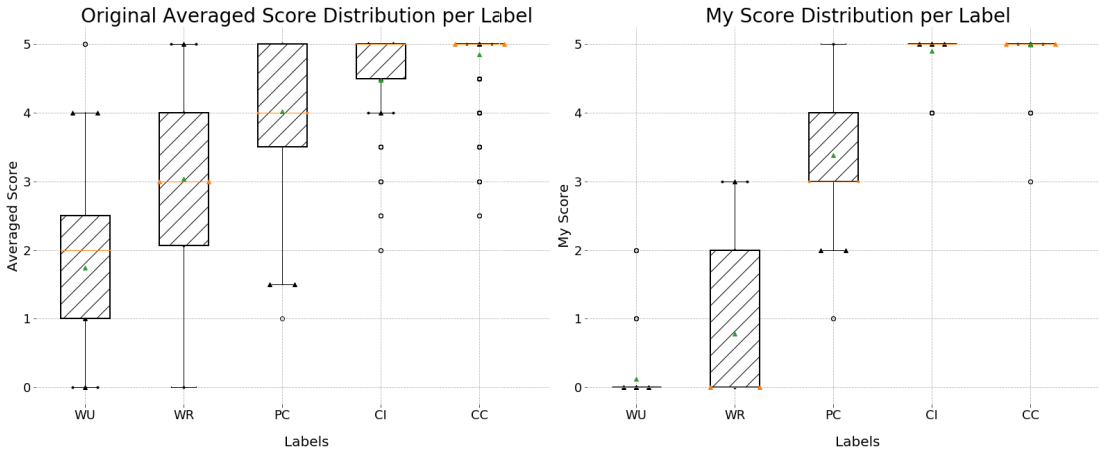

Fig. 3.4. The green dots are the mean, the red lines the median. Notice how the author's scores seperate and differentiate the classes and remove much of the overlapping inconsistency. Almost all scores classified as wrong (WU or WR) have a score value <= 2.5. This is where the failure threshold is derived for false answers in binary classification on the dataset. Also an upper threshold of 4 looks like a good choice as a low cut-off for *correct* scores.

As can be seen clearly from the author's scores shown Fig. 3.4, it makes sense to combine *correct_indirect* and *correct_complete* labels into one *correct* label since their score distributions are so similar. The same is also done for *wrong_unrelated* and *wrong_related* since although their score distributions are different, their mean and median scores are below 1 which approximately as good as failing in this dataset (for instance, a student might receive a one if his answer is completely wrong, but he does write some correct information roughly related to the topic for which the grader grants a "sympathy" point). This results in three categories, which forms the foundation for three class classification. The resulting frequency counts per class can be seen in Fig. 3.5 for three classes and the binary classification problem. The same hand-grading analysis was used to divide the data along binary classification lines as well by using the discovered threshhold of score values below 2.5 which as can be seen in Fig. 3.4 are almost always labeled as *wrong*.

The Grading Process and Score Distribution

The score distribution, in the author's opinion, has two main causes, at least for these datasets. The first, is that most short-answer responses are on topic and indeed approximately correct, for example, if a question is asking about stacks the responses will tend to have something to do with stacks and not be completely

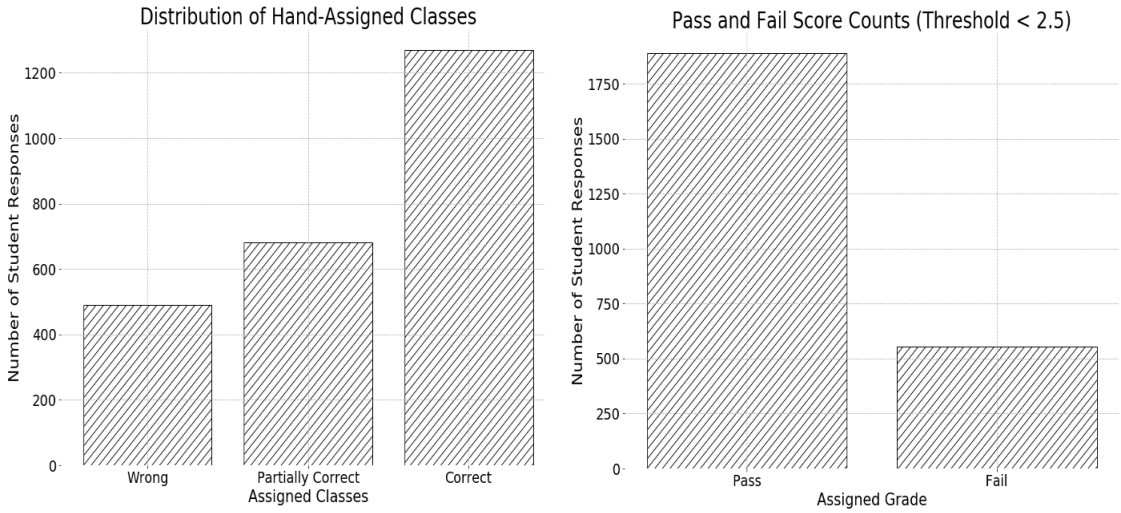

Fig. 3.5. Distribution of student grades when grouping the data by into a three class and binary classification problem based on the thresholds of $x < 2.5$ for *wrong* responses and $x \geq 4$ for *correct* responses.

unrelated (*wrong_unrelated* only makes up 20% or 1 out of every 5 wrong answer responses). The second is that the grader has a model answer and measures a student response against this ideal. This process lends itself to a sort of grading that removes points as a student response differs more and more from the model answer. It is like a sculptor removing negative space from a block. The grader compares the response to the model answer and if everything lines up the student gets a perfect score, however, for errors this score is then reduced or chiseled away by some amount depending on the subjectively interpretted severity of the mistake. These two factors explain some of the skewedness in the score distribution, in the author's opinion. It would be very interesting to see if this pattern reamerges across other short-answer topics and domains across more diverse and larger datasets.

In summary, the manual work of scoring student responses helped to clear up many aspects of the dataset, which may or may not generalize to other ASAG problems. It was especially helpful in discovering an emergent pattern of three classes, which can be further combined into two classes (pass/fail) based on discovered cut-off threshold values. From the author's perspective, grading student responses is tedious and time consuming in all but the edge cases. Maintaining consistency across different student responses and the same student responses at different points in time are two of the most difficult problems. Additionally, one pattern emerged from the scores in the dataset, namely, that three classes can be used to ease grading and improve consistency, at least from the grader's perspective, of the students' responses. Finally, the coupling of on topic student responses and model-answer based grading was given as a possible explanation for the left skewedness of the datasets.

Student Response Structures

During the process of grading, open questions were at the forefront. Some of these were, for instance, questions such as what were internal structures and patterns that kept reappearing in student responses or how did certain structures influence grading behavior. In order to analyse and attempt to answer some of these questions, the author further annotated student responses within each of the label categories in an attempt to dissect why certain responses were demoted or received less than a perfect score and even in some cases why otherwise incorrect responses received some points at all. Ultimately, there were eleven labels which student responses could receive. These labels which were used to annotate student responses in the dataset are displayed in Tab. 3.3 which shows the label, the description of the label, and the grading behavior that the author took for a student response receiving the particular label. Its also important to note that student responses could have zero or more labels associated with them.

One quite interesting structural aspect of the dataset that emerged after applying this annotation process, is that 54.55% of the *partially_correct* student responses contain either the label *Missed Concept* (28.89%) or *Partially Missed Concept* (25.66%). This means over half of the most difficult to grade student responses contained structural evidence that a student had either fully or partially missed an important concept and that partial and full misses were roughly about as common. This number jumps up even higher to 61.3% for student responses classified as *wrong*. Thus, missing concepts whether partial or complete are by far the largest factors for determining the placement and scoring of a student response. This provides some indication for the importance of researching the ability to detect and determine concept presence and absence in student responses.

Another interesting structural insight is that the *False Assertion* label can be found in 27.7% of *wrong* student responses compared to 11.6% of *partially_correct* responses and a mere 4.3% of *correct* responses. This appears to provide some evidence that making false assertions in a response has a more severe negative impact on grading than simply missing a concept or a part thereof.

Finally, , 30.4% of *partially_correct* responses contained the *Not Enough Info* label compared to only 6.3% of *wrong* responses. This indicates that almost 1 in 3 student responses do not provide enough information to clearly articulate to a human grader that they know and understand the piece of information requested in the question prompt and that when this occurs the response often receives partial credit instead of being categorically written off as false.

A full and detailed listing of the statistical distributions of each of the labels across the dataset can be found in the appendix in Tab. A.1. The author thinks this information is very useful for guiding future research and further study of short-answer responses.

Dataset Pruning

Finally, the author examined the lengths and relationships between the *(model-answer, student-response)* pairs and determined there were, broadly speaking, two

Label	Description	Grading Behavior
Missed Concept	Student failed to get one or more important concepts	Knock off multiple or all points.
Partially Missed Concept	Student missed a part of a concept or idea.	Knock off a one to a few points.
Extra Info	Student provided a significant amount of extra information which is correct but not needed.	Do nothing. Rarely grant a point or two if student otherwise missed everything.
Lack of Knowledge	Student answer is worded in such a way that it is obvious the student does not comprehend the material.	Knock off multiple points.
Misinterpretation	The student misread the question prompt or interpreted it in a way which was not intended leading to a completely false answer.	Give a few points if interpretation and response are both reasonable.
Not Enough Info	Student's answer could be correct but it doesn't provide enough description and clarity to be sure.	Knock off one to a few points.
Misspelling	Response contains a significant misspelling error.	Do nothing or knock off a point or two if the errors are extensive.
Confused Wording	Response is worded so poorly it is very difficult to assertain its correctness.	Knock off a few to multiple points.
Indirect Wording	Response is correct but wording is significantly different than model response and possibly requires inferential knowledge by the reader.	Do nothing.
No Response	Student did not answer the question or made virtually no attempt at it.	Give zero points.
False Assertion	Student states something which is completely false.	Take off a few to multiple points depending on severity.

Table 3.3. Annotated labels applied to student responses for exploring structural patterns in student responses and connected grading behavior.

types. The first type, were questions which are typically associated with ASAG, that is *(model-answer, student-response)* pairs of varying lengths all above four or five words. The second type, were questions having very short model answers containing just one or two words and often student responses just as short. These usually represented questions such as *"What are the two main methods in a stack?"* which have a model answer such as *"push and pop"*. Some had model answers as short as one or two words and many of these questions were very easy for students, contained almost all correct responses and, in the author's opinion, don't represent quality ASAG data. Therefore, it was decided to exclude these very short answer type of questions. From an implementation perspective, using another approach such as a Fill-in-the-Blank question type to model these types of question items in quizzes seems more appropriate. A visual example of the these pairs is shown in Fig. 3.6.

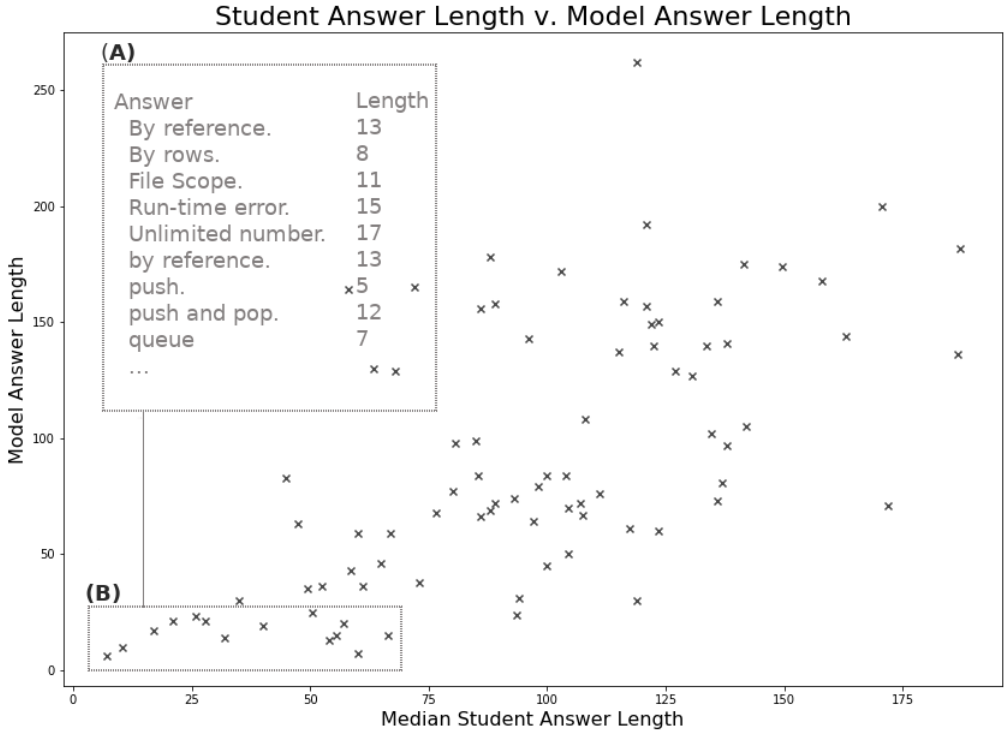

Fig. 3.6. **(A)** The textual contents of the very short model answers. **(B)** The location of the very short model answers in clustering with very short median student response lengths. Notice how all of these answers are good candidates for Fill-in-the-blank questions.

After removing these questions the dataset obtained its final size of $n = 2010$. Finally, in order to assess the usefulness of having multiple model answers during the experiments, extra model answers in two forms where also added as columns to the dataset. The first were positive model answers, that is, model answers which simply had different phrasings or wordings but should otherwise serve as a quality model answer. In addition to this type, a negative model answer was also added. This answer contained prototypically bad answers or responses which might often occur but would be deemed wrong by the human grader.

3.1.4 Translated Datasets

Since evaluating the generalizability of ASAG methods across languages is one of the purposes of this research, the main dataset (version 2.0) was translated from the original English to both German and Spanish. This was done using the google translate API, but since the author of this paper also speaks Spanish and German, some minor corrections were made to individual translations in order to touch up mistakes made by the automatic translation process. However, not all of the translations were rigorously hand editing and this could certainly be a source of error. Fig. 3.4 shows an excerpt from the dataset after translation into German.

AID	QID	SID	Prompt	Model Answer	Student Answer	Score
1	1	6	Was macht eine Funkt...	Der Name der Funktion...	Es enthält den Namen des Program...	4.5
1	1	5	Was macht eine Funkt...	Der Name der Funktion...	Es enthält die spezifischen Info...	3.0
1	1	8	Was macht eine Funkt...	Der Name der Funktion...	Die Funktionssignatur enthält de...	5.0

Table 3.4. Example showing an excerpt from the CSV file translated into German for version 2.0 of the Mohler Dataset.

The German and Spanish language versions of the datasets are used in places throughout the experiments in order to asses how and to what degree a particular methodological approach to ASAG generalizes and can be applied to other languages.

3.1.5 Other Datasets Considered

The Mohler datasets are not the only datasets available for some form of short-answer evaluation, however, they are the best the author could find for the specific problem of Automatic Short Answer Grading as defined in this thesis i.e. for assigning a numeric value to a short answer response item.

The SemEval-2013 Task 7 challenge contains a more well-balanced dataset as well as reference answers and student responses, so on the surface it looks like it would be a more apt choice. Unfortunately, the judgments or gold standard labels in this case are classifications in 5 categories: *Correct*, *Partially_correct_incomplete*, *Contradictory*, *Irrelevant*, and *Non_domain* [DNB+13]. These categories, unlike those created in the process of hand-grading in this thesis, do not translate well to a system for assigning numeric scores to short-answer questions. For instance, what value should a *Partially_correct_incomplete* item have? Should all *Correct* labeled items receive a perfect score? Are all contradictory questions wrong or are some partially correct? Without being able to group and analyze class labels according to numerically assigned grades it is extremely difficult to determine how they should be interpreted from a grading standpoint. For this reason, the SemEval 2013 Task 7 dataset is not used.

Another dataset considered was the one provided by the Kaggle hosted competition "Automated Student Assessment Prize, Phase Two – Short Answer Scoring." sponsored by The William and Flora Hewlett Foundation [Inc18]. This dataset contains human grades for short-answer response items, as well as being large (17,000 training items) and diverse, ranging over disciplines from English Language Arts to Science [Inc18]. It too appears initially to be a more than adequate dataset, however, on further inspection there are no reference answers for student responses so this dataset turns out to approach the problem of ASAG without any input from teachers and instead looks solely at a student response for determining a score. This is an approach which though interesting, is ultimately a method this paper decides not to evaluate. For this reason, the HP Dataset was not used.

Finally, there may well be other more well-balanced datasets online of which the author is not aware and did not discover in his research, though he did attempt an exhaustive search both in the literature and online.

3.2 Upper and Lower Bounds

In order to interpret the results of the experiments, it is important to have a frame of reference, a yardstick as it were, with which to inspect and reflect upon the results. This frame of reference needs to be defined for lower and upper boundary regions, that is, for the worst performance that can be expected, typically, referred to as a baseline, and for the best performance, which in the case of ASAG is human level performance.

3.2.1 Lower Bounds

One measure of poor performance a grader could give would be to look at a student response and then randomly pick a grade and assign it to that response. This is the monkey banging on a keyboard approach and randomly assigning a score to a student response. Fig. 3.7 shows the results of running the final dataset through a random scorer; these results provide a sanity-check and worst-case baseline for evaluating experiment results.

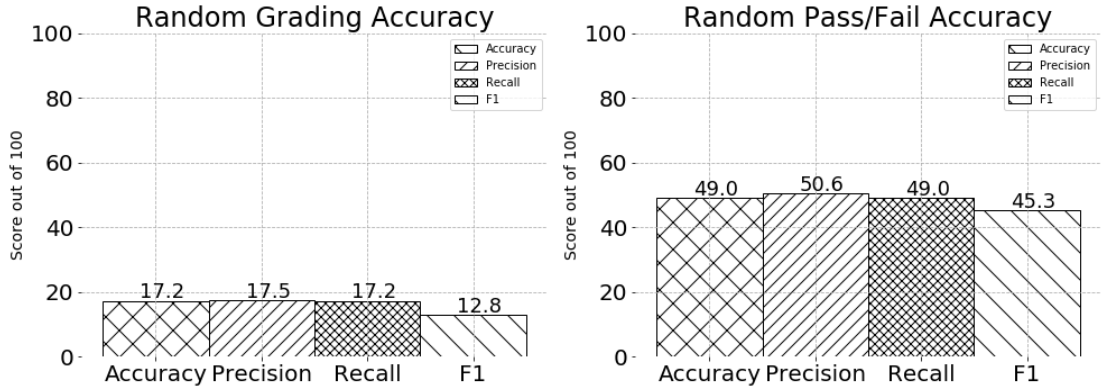

Fig. 3.7. A monkey randomly picking grades and assigning them to student short answer responses would have accuracy measurements of around 17% for continuos grading and 50% for pass/fail grading on the dataset. These statistics are derived from elementary probability theory.

.

In addition to the accuracy statistics in Fig. 3.7, both Pearson's R and Cohen's Kappa are zero as is to be expected for random sampling and the mean absolute error is 1.63 and 0.29 for the regular grading and pass/fail grading respectively.

3.2.2 Upper Bounds

The upper bounds represents the level at which a well-performing model could be reasonably expected to automatically score student responses on this dataset. The key attributes of such an ideal solution are that it would be roughly equivalent to a human grader. Further, this grader should be a domain expert on the topic as well

as unbiased, that is, the human expert must not know or be capable of associating students with their respective scores. This last attribute mitigates against some of the problems with grading as already discussed in the introductory chapter. To calculate the statistics for such a model, it seems reasonable to choose the author's own hand-graded performance measured against the gold standard scores to determine this upper boundary. The author of this thesis is a professional software engineer and thus a domain expert for topics in introductory computer science classes, having even taught such a class once himself. Additionally, he has no way of being biased against the students in the dataset since he does not know them or know their names or anything else about them. So, in order to calculate the measurements for an upper performance boundary, the author's hand-graded scores were compared to the gold standard scores and these upper limit measurements were calculated for the 6-class score assignment problem, 3-class assignments as well as pass/fail grading. The results are shown in Tab. 3.5.

Measure	**6-Class Grading**	**3-Class Grading**	**2-Class Grading**
Accuracy	47.21%	72.69%	81.84%
Precision	57.09%	78.32%	82.63%
Recall	47.21%	72.69%	81.84%
F1	45.95%	73.85%	78.06%
R	0.70	0.65	0.42
Kappa	0.26	0.44	0.35

Table 3.5. Comparing the author's scores vs. the other human graders scores on the dataset. Shows key measurements for grading into 6 Classes, 3-Class Grading (Wrong / Partially Correct / Correct) and binary classfication i.e. Pass/Fail grading.

Tab. 3.5 shows the upper bounds of expected performance from a good model and represents values which are considered at the upper end of potential performance for any given model using this dataset.

3.3 Methods

3.3.1 General Preprocessing

Although preprocessing does not comprise its own experiment category, understanding the methods used to preprocess textual data is important for assessing every subsequent phase of an experiment. Preprocessing techniques are designed to remove noise and prepare data for later stages in each experiment's pipeline.

Stopword Removal

Stopword removal is the process of removing common grammatical and function words from text because such words do not convey much information [MS08]. For instance, the word *the* occurs all the time in English text and although it can have an important semantic function, its information content is usually very low, especially in the context of searching for word-by-word matches between sentences [MS08]. For example, if a user enters *"the red dress"* into a search box on a website, *"the"* is probably contained in every document and is therefore useless, however, the words *"red"* and *"dress"* are likely only contained in documents related to dresses or the color red, therefore, these words have much higher information content. Code Listing 3.1 shows exactly how stopword removal works.

Listing 3.1. Stopword Removal Example in Python using NLTK. Notice how the function words *A*, *is*, *in*, *to*, *for* and *the* are removed from the final result.

```
# import list of standard stopwords from NLTK
from nltk.corpus import stopwords
english_stopwords = stopwords.words('english')

# student response
sa = ("A prototype program is used in problem solving"
      " to collect data for the problem.")

# remove stopwords
result = [word for word in sa.lower().split(" ")
          if not word in english_stopwords]

print(result)
# ['prototype', 'program', 'used', 'problem',
#  'solving', 'collect', 'data', 'problem.']
```

Case and Whitespace Normalization

Often it is helpful to ignore differences in word orthography due to case, for example, a task would like to treat *the*, *The*, and *THE* as the same token. Normalizing uppercase and lowercase differences of tokens in a text stream to a standard (either all letters in uppercase or all in lowercase) is a process known as case normalization [MS08]. In this thesis, all tokens were transformed to lowercase before further processing steps. Notice that although this method resolves some issues, such as not recognizing the words *Code* and *code* as equivalent, it can create other problems by removing key distinguishing characteristics of proper names, for instance,

making the word *brown* in the phrases *Max Brown* and *brown paint* indistinguishable. Additionally, and often in tandem with this step, whitespace characters are removed from the beginning and end of the input string and multiple spaces and line separators are merged into one. Listing 3.2 shows a code example for both of these preprocessing tasks.

Listing 3.2. Case and Whitespace Normalization Example

```
import re
# student response with wierd spacing
sa = "  A Run-time    error. "

# normalize whitespace
res1 = re.sub(r'\s+', ' ', sa.strip())
print(res1) # 'A Run-time error.'

res2 = res1.lower()
print(res2) # 'a run-time error.'
```

As a final note, case normalization is also applied to other languages explored in this thesis, namely German and Spanish, however, it is irrelevant for languages whose orthographic script does not have the concept of case such as Chinese or Arabic. These are, however, outside the scope of this thesis.

Stemming

Stemming is a mechanical, algorithmic process which strips off affixes and leaves word stems remaining [MS08]. It represents one way to normalize the morphology of words. For instance, stemming may convert the words *lovely*, *loved*, *loves* all to the stem *love*. Its usefulness in the case of Information Retrieval (IR) is debatable [MS08], however, it is applied in this research when looking purely at surface lexical structures since any other form of morphological normalization would subtly introduce semantic knowledge into the process. A code example using NTLK is provided in Listing 3.3.

Listing 3.3. Stemming Example

```
from nltk.stem.snowball import SnowballStemmer

# create a stemmer following the Snowball algorithm
stemmer = SnowballStemmer("english")

sa = "jumps jumping jumped jumper".split(" ")

print([stemmer.stem(token) for token in sa])
# prints ["jump","jump","jump", "jumper"]
```

Lemmatization

Lemmatization is the process of trying to find the underlying lexeme or lemma of a word given its inflected form. This technique implies having syntactical and semantic knowledge of the words within their context. For instance, the word *lying* could be the verb *lie-lay* meaning "to put something down" or *lie-lied* meaning "to fib" [MS08]. Listing 3.4 shows an example which uses the SpaCy natural language processing library.

Listing 3.4. Stemming Example

```
# import and load the spaCy library for processing text
import spacy
nlp = spacy.load("en")

sa = ("it simulates the behavior of portions"
      " of the desired software product")

# perform lemmatization
res = nlp(sa)

# extract the lemmas
print([t.lemma_ for t in res])
# ['it', 'simulate', 'the', 'behavior', 'of',
#  'portion', 'of', 'the', 'desire', 'software', 'product']
```

Punctuation Removal

In many contexts punctuation is not very useful. For instance, commas and periods may not be useful to a machine in a task that is not text-to-speech. The punctuation removal step gets rid of periods, commas and other characters such as semicolons and exclamation points which can introduce noise by preventing algorithms from recognizing *"success"* and *"success!"* as the same word for example.

Unexplored Preprocessing Techniques

There are several other preprocessing techniques which could improve performance of some of the approaches tested in this thesis. For instance, approximately 3% of the student responses contained significant spelling and grammatical mistakes according to the author's hand-labeled annotations. Applying automatic spell-correction as a preprocessing step could potentially mitigate some subset of these errors and boost overall performance. This technique, however, was neither applied nor explored.

Another technique which was not explored is known as *Pronoun or Anaphora Resolution*. It resolves references to the same entity or thing. For example, given the question prompt, "What is a variable?" and a student response "It is a symbol or name for a value or number.", this technique can recognize the noun phrase *It* as referring to the noun *variable* and, consequently, replace *It* with the word *variable*. This can be useful to avoid ambiguity, especially for questions which expect short-answer responses spanning multiple sentences in length.

3.3.2 Regular Expressions

Regular expressions, also called *regex* or *regexp*, are used in the experiment category of the same name. Regular expressions have their own notation which is similar to a simple programming language, and they allow the user to describe, parse and extract text and text segments [Fri08]. For example, imagine a human grader wishes to detect whether or not a student response contains the words *enqueue* and *dequeue*, perhaps the question is asking about methods available for working

with a queue. The human grader could write a regular expression, for instance, $((en|de)que(ue)?)$ which, when applied to a student response string, for example: *"The methods enqueue and dequeue are used for working with queues."* would return the matching words *enqueue* and *dequeue*. A complete example of the above scenario in Python code is shown in Listing 3.5. Additionally, that same regex would, in this particular case, also match against the commonly misspelled variants enque and deque, allowing for the teacher to more flexibly find matches even when those words have been misspelled by students.

Listing 3.5. Usage and Application Example for Regular Expressions in Python Code

```
1  import re
2
3  text = "The methods enqueue and dequeue are used when working with queues"
4
5  # define the rules of the language
6  rules = "((:?en|de)que(:?ue)?)"
7
8  # compile the rules into a FSM
9  regular_expression = re.compile(rules)
10
11 # apply the string as input to the FSM to discover all matching substrings
12 result = regular_expression.findall(text)
13 print(result)
14 # prints ['enqueue', 'dequeue']
```

The crux of the way regular expressions work is by defining a set of symbols and letters, some of which can be seen in line 6 of Listing 3.5, which allow a human to then design a regular language using these symbols. A finite state machine (FSM) is then automatically generated for the designed language. Now, any new input string can be run through the FSM in order to detect whether or not it is a valid string for the language. For instance, the regex $((en|de)que(ue)?)$ defines a language of all strings which start with either *en* or *de*, are followed by the three letters *que* and optionally have the suffix *ue* (the question mark is a symbol which means that the preceeding character sequence is optional). As a string is parsed, its input is fed into the FSM, if the input is a valid string for the language, for instance, *en* then the state machine moves along to its next state and processes input again, repeating this process until either it receives an input which is not part of the language, at which point it exits, or it reaches the final state at which point it knows it has read a valid string for the language and can capture and return that string.

There are numerous complexities and subtleties to regular expressions and entire books which have been written on the topic. This section serves only to give the reader an approximate idea and orientation as to what is occuring during experiments which use regular expressions.[2]

[2] *Mastering Regular Expressions* by Jeffrey Friedl is recommended for those wishing to understand the complexities of regular expressions in detail.

3.3.3 Lexical Structure

The techniques used for analyzing lexical structure are carefully crafted to look only at the surface structures of model answers and student responses. The first set of techniques in this category are word overlap methods. These methods view each word as an opaque symbol called a token and the sentence long responses as unordered sets of these tokens. Set operators such as union and intersection are used and, importantly, word frequency counts play no role in set-based approaches since when viewing a sentence as a mathematical set a token or member element may only occur once.

Word Overlap Equations

The idea of using word overlap measurements for ASAG, and specifically, the dice, jaccard, and cosine coefficient equations, are taken from the paper *"Automatic Short Answer Scoring Using Words Overlapping Methods"* by Pribadi et al [PAP+17].

Dice Coefficient

$$\frac{2|X \cap Y|}{|X| + |Y|}$$

This method takes the intersection of the words in two sentences X and Y and multiplies it by 2. It divides this value by the number of words in the first sentence plus the number of words in the second. If the two sentences have no words in common then the similarity score will be zero. If they are exactly the same the score will be 1. Thus this method, as well as all the other word overlap methods explored, has a range of $[0, 1]$.

Jaccard Coefficient

$$\frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

The jaccard coefficient looks at the total words which X and Y have in common, normalized by the total words in both sentences after subtracting the number of words they share. Its range is $[0, 1]$ and it generally gives more conservative i.e. lower scores than the *dice coefficient* since it subtracts the words in common within the denominator instead of multiplying the intersection size by two in the numerator as the dice coefficient does.

Cosine Coefficient

$$\frac{|X \cap Y|}{|X|^{1/2} \cdot |Y|^{1/2}}$$

The cosine coefficient measures the ratio of the number of shared words over the square root of the product of the length of the words in each sentence.

Faucett Coefficient

This is a custom coefficient adapted from the cosine coefficient. The idea is to take a power greater than 0.5 and less than 1 of the ratio between the overlapping words and the combined words of both sets. Taking the power means that relatively low overlap scores like 0.1 will be scored more highly: $0.1^{1/2} = 0.31$ for example. This can be thought of as performing score boosting or being more lenient in similarity scoring. Also by putting the union of both sentences' words in the denominator, there is still a penalization for longer sentences which have less overlap.

$$C(X, Y, p) = (\frac{|X \cap Y|}{|X \cup Y|})^p, \ where \ 0.5 \leq p < 1$$

Finally, a comparison of these methods can be seen in Fig. 3.8.

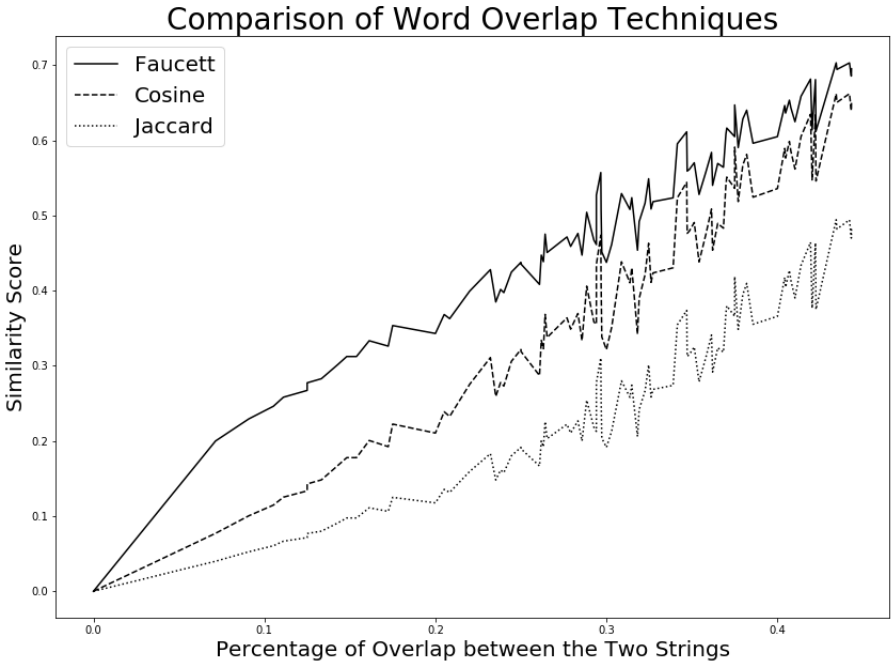

Fig. 3.8. To generate this image, random sentences were generated and then sorted ascending by the number of overlap words they shared. The word overlap methods' scores were calculated and ploted to observe how they behave as the word overlap between sentences increases. Notice how the jaccard is most conservative and faucett most lenient in scoring two sentences for similarity. Also apparent is the fact that the custom faucett method boosts lower scores, whereas for higher similarity scores the faucett and cosine coefficient scores are not very dissimilar. Note that the dice and cosine coefficients were so similar that the dice coefficient was left out. The difference between these two was on the order of hundredths of a percentage point and invisible in the graphic.

Bag-of-Words (BoW) and N-Grams

Many experiments make use of n-grams. N-grams are ordered groupings of words where n is the number of words [MS08]. For instance, the given sentence *"A variable is a name for a value"* when split into bigrams (n-grams where $n = 2$) forms the set of values *[(a, variable), (variable, is), (is, a), (a, name), (name, for), (for, a), (a, value)*. These n-grams can then be used to compare two sentences and see how many n-grams they share, the greater the number of shared n-grams the more similar the two sentences are interpreted to be. This is often done using Bag-of-Words techniques (BoW), which converts sentences into a matrix representation where the columns are the sentences, the rows are the words or n-grams and the values in the cells are the word occurance counts in the sentences. A distance or similarity score for two rows can be calculated using cosine simlarity of the row vectors.

N-grams can be created for multiple values of n per sentence, for instance, storing bigrams, trigrams and even 4-Grams for a given sentence. However, as n increases, it becomes increasingly less likely that a match between n-grams of two different sentences will be found. This is because it is less likely to have four words in the exact same sequence in two different sentences than it is to have three words in the exact same sequence and three is more likely than two, etc. This effect is only compounded when stopwords are removed. In order to compensate for the lower probability larger n n-gram matches, the proportion of matches to total such n-grams can be raised to a power of $1/n$, which grants a higher score for a lower probability match. Thus a trigram match between two sentences each containing three trigrams would have a score of $s = (\frac{1}{3})^{1/3}$. Fig. 3.9 illustrates this point and shows a detailed example of how the probability of finding n-gram matches of increasing size decreases with the length of the n.

This same approach is also used on Bag-of-Words methods which take larger n-grams into account.

3.3.4 Syntactical Structure

The methods and techniques used to examine syntactical structure are designed to extrapolate information about sentence similarity from the ordered relationships among words. For instance, the sentence *"John gave Sally the ball."* and *"Sally gave John the ball."*, although containing the exact same words have different agents carring out different actions and, consequently, different meanings. This difference is based exclusively on the ordering of words, particularly the words *John* and *Sally*, in the sentence.

Edit Distance

The edit distance is a general measure for figuring out how similar two strings are, specifically by looking at the ordering of the members of each string. It measures order and can be thought of as measuring how difficult it is to align two strings. Formally, the minimum edit distance is defined as the minimum number of deletes,

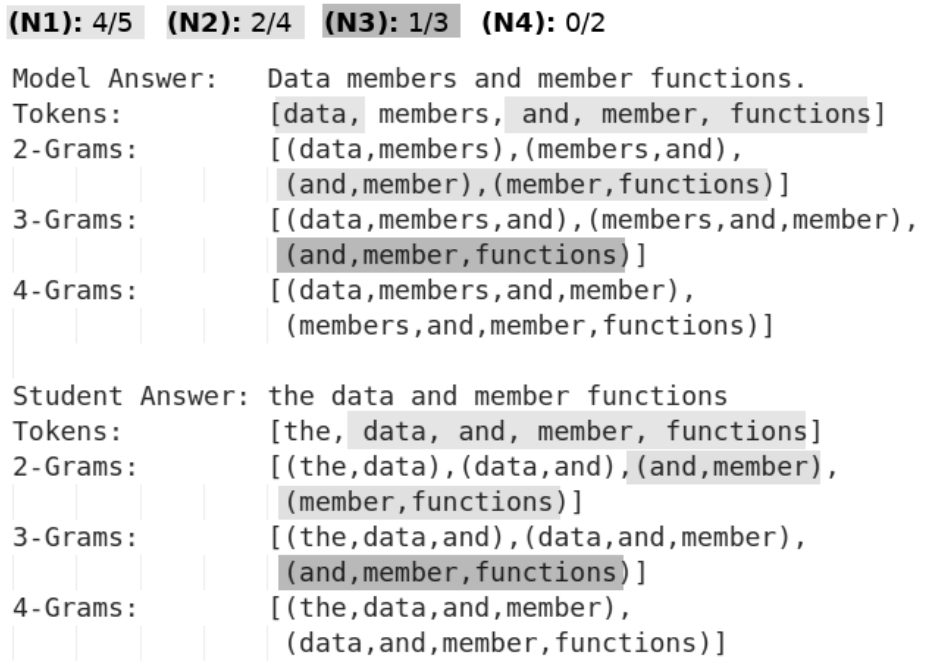

Fig. 3.9. Note how the tokens **(N1)** have a matching score of 0.8, the bigrams **(N2)** a score of 0.5, the trigrams **(N3)** a score of 0.33, and there are no 4-gram matches **(N4)**. This shows why n-gram matches of greater n should have some form of score boosting; it is simply the case that they are much less likely to occur and hence carry more information content.

insertions, substitutions, that is editing operations, needed in order to transform one string into another [Jur18]. Usually, the edit distance is performed on strings in order to compare words but it can just as easily be used to compare the structure and order of sentences and this later use is the preferred application in this thesis. Fig. 3.10 shows visually how the minimum edit distance process can work at a sentence level.

Each of the operations as seen in Fig. 3.10 can also receive a penalty cost. Typically each operation thus has a penalty cost of 1, which means that two sentences which are exactly equal have an edit distance of 0 and those which share no words have an edit distance which is the length of the longest sentence. This can be generalized into an edit similarity score with a range of $[0, 1]$ where 1 is a perfect matching sequence and 0 symbolizes no ordered similarity. The full equation is given in Eq. 3.1.

$$Edit\_Similarity(S_1, S_2) = 1 - \frac{Edit\_Distance(S_1, S_2)}{max(|S_1|, |S_2|)} \tag{3.1}$$

What is interesting about the edit distance is that it can be adapted to find the cost of transforming from any vectorized representation to another, not just among words or sentences. For instance, a dependency parse of the syntactical relationships between words can be made and and stored in an ordered vector,

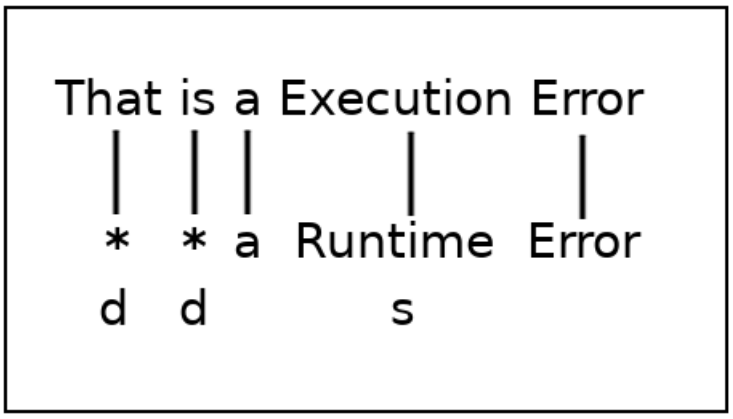

Fig. 3.10. Representing the minimum edit distance between two sentences as alignment. The final row gives the operations for each action needed to align the two sentences. **d** = delete, **s** = substitute. Not shown in this example is the insert operation.

then the edit distance can be calculated for transforming from one dependency parsed vector representation into another. The same can be done for part-of-speech (POS) tags between two sentences.

Bleu Score

The Bleu score is a method developed in order to evaluate the quality of the results of automatic machine translation of texts. It generates n-grams in order to compare a candidate translation and multiple reference translations, it weights these n-grams, computes their geometric average, penalizes sentences of longer length and has a range of $[0, 1]$ with 1 representing identical sentences and 0 sentences which have no similarities [PRWZ01]. It also has a way for penalizing repetitions of words, which in the context of ASAG, would prevent a student from acquiring better scores by simply repeating some small phrase which they know to be in the model answer, for example, entering the bigram *function signature* five times in response to the question prompt *"What are the components of a function signature?"*. This prevents rewarding a student response which simply generates an abundance of resonable words.

Additionally, it has been shown to correlate highly with human assessment of translated sentences [PRWZ01]. In terms of looking at sentence structure, the Bleu score is interesting because it takes into account word order, sentence length, and overall sentence structure. An example of the Bleu calculation in python code is given in Listing 3.6.

Listing 3.6. Bleu Score Example in Python using NLTK

```
1 # import the bleu score for sentences from NTLK
2 from nltk.translate.bleu_score import sentence_bleu
3
```

```
# create a reference sentence
reference = [['the', 'quick', 'brown', 'fox',
              'jumped', 'over', 'the', 'lazy', 'dog']]

# create a student input sentence
candidate = ['the', 'fast', 'brown', 'fox',
             'jumped', 'over', 'the', 'sleepy', 'dog']
score = sentence_bleu(reference, candidate)
print(score) # prints 0.485
```

Tree Edit Distance

The Tree Edit Distance between two trees can be defined as the minimum cost of edit operations which are needed in order to transform one tree into another. Of the various edit operations which are possible, the most common are inserting a node into a tree, deleting a node and changing the label of node [SGAM+15]. Fig. 3.11 demonstrates exactly how this a dependency parse tree structure is created from two sample sentences and how one sentence is transformed into another using insertions, deletions and label changes.

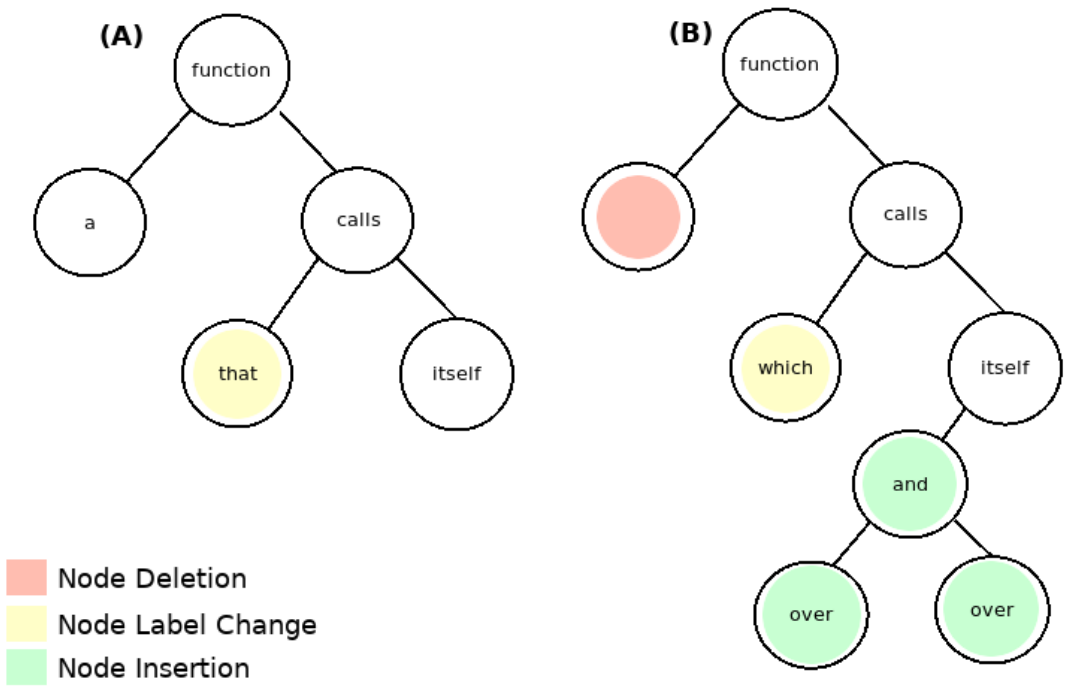

Fig. 3.11. **(A)** the model answer is transformed into **(B)** the student response. Each tree edit carries an associated cost with it, and thus the distance between two sentences can be quantified as the size of this cost. Notice how two identical sentences will have a cost of 0 and sentences which are structurally very different will have higher costs.

These distance metrics can be turned into similarity scores using Eq. 3.2 [SGAM+15].

$$\begin{aligned} sim_1(T_i, T_j) &= \frac{1}{1+d} \\ sim_2(T_i, T_j) &= \sqrt{\frac{1}{1+d}} \end{aligned} \tag{3.2}$$

where T_i and T_j are trees and d is the tree edit distance between T_i and T_j. As can be seen the tree edit distance provides another means of quantifying the ordered relationship between two sentences and for measuring the syntactical similarities between model answers and student responses.

3.3.5 Semantics

Knowledge and Corpus Based Word Meaning

There are, generally speaking, two ways to capture the meanings of words in texts: statistical methods or human crafted knowledge bases. The latter is usually realized as hand constructed dictionaries, in which one can identify word sense by looking up a lexeme and finding synonyms and synonyms of those synonyms, etc. which leads to the discovery of an entire network of meaning encoded within the structure of the dictionary. This is the idea behind *WordNet*, which is an electronic dictionary of English, having words organized into graph hierarchies. Each node has a *synset* (sense set) of associated words which have identical or closely related meanings. Besides synonymy, other types of relationships between words are also encoded in *WordNet* such as meronymy (part-whole relationships) or hyponymy (type-of relationships) [MS08]. The English version of *WordNet* can be downloaded for free from the internet [MS08]. However, this is not the case for the German version known as *GermaNet* which requires a license [oT18]. Thus, although there are versions of *WordNet* in other languages such as German and Spanish, due to the amount of manual labor involved in creating robust dictionaries among other factors, there is unfortunately a high degree of variance in performance as well as availability, especially when one considers rarer or non Indo-European languages.

The other approach to understanding word meaning is to algorithmically analyze tons of corpora and discover statistical similarities between words and thereby deduce which words are related to one another and in what manner. This is known as the corpus based approach.

The following sections elaborate and explain specific measurement techniques that use either one or both of these approaches. These techniques are examined in detail in one experiment in the research portion of this thesis.

Shortest Path Similarity

The shortest path method is shown in Eq. 3.3 and demonstrates a simple way of measuring the relatedness or semantic similarity of two words. The shortest path is defined as

$$Sim_{path} = \frac{1}{length} \tag{3.3}$$

where *length* is the length of the shortest path between two concepts in the *WordNet* graph hierarchy using node-counting. Intuitively, this means that the further away two concepts i.e. words are from one another in the graph the lower their similarity score will be. Conversely, two words which are very similar will have a short length and higher similarity score.

Leacock and Chodorow Similarity

The Leacock and Chodorow similarity shown in Eq. 3.4 is a knowledge based approach to word sense identification which utilizes *WordNet*. The distance between two words a and b using this algorithm is defined as

$$Sim_{ab} = max(-log\frac{Np}{2 * D}) \tag{3.4}$$

where Np is the number nodes in the path from a to b and D is the maximum depth of the taxonomy [LC98]. Notice that length is measured in nodes and not links, thus Np for two sibling nodes would be 3 (one for each sibling node and one for the parent). The length of the path between members of the same synset is 1. The depth D of *WordNet* varies depending on whether one is finding the similarity between verbs or nouns, but assuming a depth D of 14, this equation assigns words which are far apart in the *WordNet* hierarchy lower scores and words which are closer together higher scores.

Wu and Palmer Similarity

The Wu and Palmer similarity measures the depth of two given concepts in the *WordNet* taxonomy in addition to the depth of the least common subsumer (LCS) and combines these measurements into a similarity score shown in Eq. 3.5 [WP94]. It is defined as

$$Sim_{wup} = \frac{2 * depth(LCS)}{depth(concept_1) + depth(concept_2)} \tag{3.5}$$

where the LCS is the most specific concept which is an ancestor of both $concept_1$ and $concept_2$ [WP94]. An detailed example of this approach is shown in Fig. 3.12 for finding the similarity between the two concepts *Car* and *Boat*.

Intuitively, words which are very close in meaning, for instance *dog* and *canine* will have a LCS with a high depth and their respective value will approach one. For words whose LCS is the root of the wordnet node hierarchy they will have a similarity of 0. Because this method uses the depth of the graph hierarchy and the LCS, it incorporates type-of semantics into the relational meaning between words returned by its similarity score.

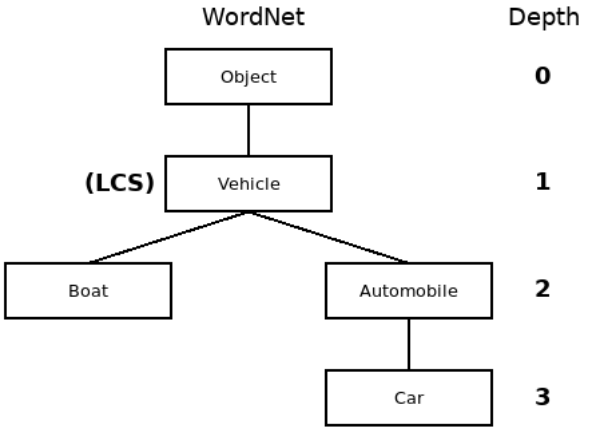

Fig. 3.12. Demonstration of the Wu and Palmer similarity for the two concepts *Car* and *Boat*. The concept *Vehicle* is the Least Common Subsumer **(LCS)** of both *Car* and *Boat*. The depths within the graph hierarchy are given on the side, where *Car* has a depth of 3, *Vehicle* 1, and *Boat* 2. The similarity is thus: $\frac{2 \cdot depth(LCS)}{depth(Boat)+depth(Car)} = \frac{2 \cdot 1}{2+3} = 0.4$.

WordNet and Information Content Similarity Measures

The Resnik, Lin, and Jiang and Conrath similarity all introduce a new form of similarity measure. These approaches measure the semantic similarity or distance between words and concepts by combining the information within the taxonomic structure of *WordNet* with corpus based statistical information in order to better quantify the semantic distance between nodes by additionally incorporating evidence computational derived from a distributional analyses of text corpora [JC97]. These methods use the idea of Information Content (IC) which is defined as the negative log of the probability of a given concept (based on observed frequency counts) as

$$IC(c) = -log(P(c)) \tag{3.6}$$

All these measures take as inputs two concepts c_1 and c_2 and also rely on the least common subsumer (LCS) explained in the Wu and Palmer similarity measurement. The LCS is the most specific concept that is the shared ancestor concept of two concepts c_1 and c_2 [Ped10].

The Resnik similarity uses the Information Content of the LCS and is defined as

$$res(c_1, c_2) = IC(LCS(c_1, c_2)) \tag{3.7}$$

It is a rough and fuzzy similarity measure since many rather different pairs of concepts may share the same LCS. It can be thought of as finding the more general concept which is more likely to have higher observed counts and thus a non-zero score.

This idea of fuzziness leads to both the Lin and Jiang and Conrath measures as they attempt to increase the precision and reduce the coarseness of the Resnik

measure [Ped10]. These measures can be thought of as extensions of the Resnik measure and are defined as

$$lin(c_1, c_2) = \frac{2 * res(c_1, c_2)}{IC(c_1) + IC(c_2)} \tag{3.8}$$

$$jcn(c_1, c_2) = \frac{1}{IC(c_1) + IC(c_2) - 2 * res(c_1, c_2)} \tag{3.9}$$

Word Embeddings

Word embeddings model word meaning as vector representations and take a different and, in some ways, more flexible approach to understanding meaning than human created hierarchical graph structures like *WordNet*. In this approach, words are learned by analyzing text corpora and are represented as vectors in a n-dimensional vector space where similar words are closer to one another than dissimilar ones [GPJ18]. They have been shown to have benefits over Bag-of-Words models or one-hot encoded schemes which suffer from sparse vector problems due to the sizes of the vocabulary. Further, word embeddings use the context of surrounding words to encode meaning into a dense vector representation which alleviates the sparse problem. Succinctly, a word embedding is the coordinates of a word projected into a newly learned semantic vector space [GPJ18]. Fig. 3.13 visualizes these word vectors as points in a 3-Dimensional space.

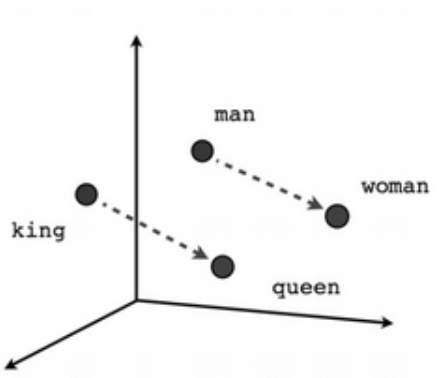

Fig. 3.13. A visual representation of word embeddings in 3D space. Notice how man and woman share similar positions as well as king and queen. This is the idea behind word embeddings: capturing distributional semantics within a vector representation.
Source: https://www.tensorflow.org/tutorials/representation/word2vec

There are different methods for computing word embeddings, two which are evaluated in this thesis are Word2vec and FastText. Word2vec representations are learned by using a neural network which is fed all the words in the vocabulary in chunks determined by a window size parameter. This neural network then attempts

to predict the probabilities of an output vector of words given the current window size it is observing and its own internal weights. After training on a large text corpus, the input and output layers are discarded and the hidden layer is used as the dense representation of word meaning [MCCD13].

FastText in many ways extends the Word2vec approach by taking into account not only words but all the possible n-gram combinations of characters within words, arguably allowing it to incorporate the semantics of plurality and prefixes and suffixes as well as different parts of speech [BGJM16].

After learning word embeddings, two words can be compared for their similarity by finding their cosine distance in the n-dimensional vector space. This is more flexible than knowledge based approaches since it allows for the semantics of any language to be learned in an automated way. It also allows for the learning of domain specific semantics, for instance, by learning word embeddings on medical or biological documents as opposed to general news articles. One drawback of this approach, however, is that large text corpora are needed and a significant amount of training time is required in order to learn the distributional semantics. In this thesis, a wikipedia dump in the given language is used for learning word embeddings for that language.

Word Mover's Distance (WMD)

A final measure of meaning which is assessed in this thesis is Word Mover's Distance (WMD), which is a function that computes the distance between two documents or texts - in this thesis student response and model answer - and is built off of word embeddings. It calculates the minimum amount of distance word embeddings in one document need to travel in order to place themselves into the n-dimensional positions of word embeddings from a second document. Fig. 3.14 demonstrates visually how one can think about this transformation given two documents d_1 "Obama speaks to the media in illinois" and d_2 "The President greets the press in Chicago" [KSKW15].

For this thesis, Word Mover's Distance is transformed into a similarity metric labelled Word Mover's Similarity and given in Eq. 3.10 and defined as

$$WMS(M, S) = 1 - \frac{WMD(M, S)}{MAX\_WMD} \tag{3.10}$$

where M is a model answer, S is a student response $WMD(M, S)$ is the calculated Word Mover's Distance result and MAX_WMD is the maximum value a vector can travel to transform itself into another vector in the n-dimensional vector space. Essentially, this means a student response identical to the model answer will receive a score of 1.0 and a response maximally far away from the model answer will receive a score of 0.

3.3.6 Scoring Equation

All of the previously discussed semantic measurements operate at a word level. However, in the context of ASAG, sentence level understanding is required and

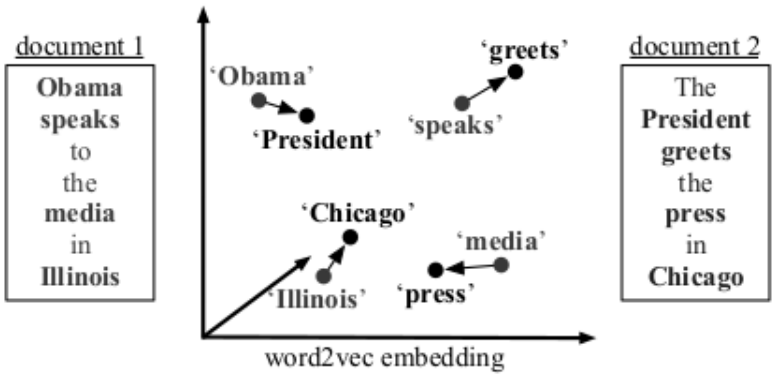

Fig. 3.14. Notice how the word embeddings from *docuemnt 1*: *Obama*, *speaks*, *media* and *Illinois* have to travel i.e. move to transform into the positions of semantically similar words in *document 2*. Quantifying the distances covered by this movement is the idea behind WMD [KSKW15].

therefore a function must be developed to evaluate phrase and sentence similarity in order to compare the sentences that make up the *(model-answer, student-response)* pair. The general equation for pair evaluation is given in Eq. 3.11 and is adapted from [MM09]. Each word in the model answer is paired up with each word in the student response and the pair with the maximum similarity is obtained. More formally, for each word m in model answer M the $maxsim$ is found as

$$maxsim(m, S) = max(SIM_x(m, s_i)) \tag{3.11}$$

where s_i is a word in the student response S. After this phase, the comparison between the *(model-answer, student-response)* pair is represented as a vector of maximum similarities, these similarities can then be summed and normalized by the length of the model answer M, the length of the student response S, or an averaged length of both. These variations are defined as

$$score(M, S) = \frac{1}{|M|} \sum_{i=0}^{|M|} maxsim(m_i, S) \tag{3.12}$$

$$score(M, S) = \frac{1}{|S|} \sum_{i=0}^{|M|} maxsim(m_i, S) \tag{3.13}$$

$$score(M, S) = \frac{2}{|M| + |S|} \sum_{i=0}^{|M|} maxsim(m_i, S) \tag{3.14}$$

As long as the SIM_x function returns a value in the range $[0, 1]$, all other functions will do so as well, allowing the result to be easly scaled to any numerical grading system using the function

$$scale(x, G) = \lfloor x \cdot G \rceil \tag{3.15}$$

where x is the calculated result of $score(M, S)$ and G is some scale for grading, for instance, 100 for a test worth 100 points, the result of the product of x and G is then rounded to the nearest integer to complete the transformation.

3.3.7 Machine Learning

The machine learning experiments are conducted using three well-known models already implemented by the scikit-learn python library [PVG+11]. Superficial details as to how these models work are covered in the subsequent subsections. For a more in-depth perusal, the interested reader can consult the scikit documentation at `https://scikit-learn.org/stable/documentation.html`.

Multi-layer Perceptron

The Multi-layer Perceptron (MLP) is a supervised learning algorithm. It can learn a non-linear function approximation for either classification or regression. Given a set of input features $X = x_1, x_2, ..., x_n$ it predicts an output variable y. It can also have any number of non-linear layers between the input layer and the final output layer. These layers are referred to as hidden layers. Fig. 3.15 shows an image of a Multi-layer Perceptron with just one hidden layer. Some of the key advantages MLPs are their ability to learn and model non-linear functions and their on-line learning capabilities [PVG+11].

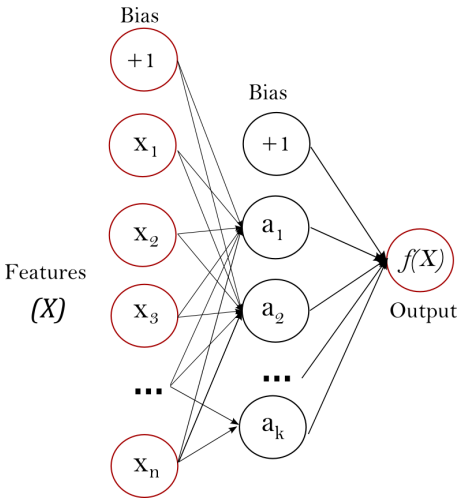

Fig. 3.15. A Multi-layer Perceptron $f(X)$ applied to the features X with one hidden layer. The weights $(a_1, ...a_k)$ are adjusted by the algorithm in order to predict the target class.
Source: `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`

Logistic Regression

Logistic Regression, particulary the scikit-learn implementation, is another learning algorithm used in this thesis to attempt to model the underlying process of short answer grading. The Logistic Regression implementation uses the sigmoid function $sig(x) = 1/(1 + e^{-x})$ to estimate the probability that a given set of input features corresponds to a particular value of the target variable, where the target variable is usually binary but can be extended to multiple classes.

Random Forest

A random forest is the final learning algorithm assessed. It trains decision tree classifiers on subsamples of the data and uses averaging to improve the predictive performance of the model [PVG+11].

3.4 Statistical Evaluation Measures

This section details statistical techniques applied for evaluating experiment results as well as the reasoning behind the particular choice of technique in the context of ASAG.

In the dataset there are gold standard scores which have been assigned to each student response and in the experiments the developed models attempt to predict these scores given each student response item. This means that after running an unsupervised approach, there is an array of size n, ($n = 2442$ in the case of v2.0 of the dataset) containing tuples of the form $(s_{true}, s_{predicted})$ for each student response in the dataset. This does not hold for the final experiment, which is unsupervised and uses five-fold cross validation to score the results.

The following subsections outline various approaches to understanding what the list of score tuples can tell about the underlying model used to run the experiment.

3.4.1 Predictions: Accuracy, Precision, Recall and F1

In order to conceptualize the various metrics in this section it is helpful to have a reference confusion matrix of a binomial experiment, for instance, an ASAG problem where the model predicts simply *pass* or *fail*. This confusion matrix for this is shown in Fig. 3.16.

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Fig. 3.16. A confusion matrix demonstrating a model's predicted values (Predicted Class) vs. the true values (Actual Class).

The **True Positives (TP)** are correctly predicted positive values. For instance, when the model runs, these are values for which the tuple $(s_{true}, s_{predicted})$ is scored as $(pass, pass)$. Accordingly, **True Negatives (TN)** are correctly predicted negative values, having tuples of the form $(fail, fail)$. **False Negatives (FN)** are values for which the gold standard value is positive i.e. *pass* but the model predicted *fail*, these have the tuple form $(pass, fail)$. Finally, there are the **False Positives (FP)** which make up those instances where the true value was negative yet the model predicted positive i.e. $(fail, pass)$.

Accuracy

Now, having an understanding of the possible prediction scenarios the model can make, the evaluation metrics can be explained. The first metric anyone would likely think of is accuracy, and for good reason, it is a useful metric that is easily

interpreted. Accuracy is simply the ratio between the number of correctly predicted observations to the total observations. Or to put it in the terminology of the previous paragraph, it is the ratio of True Positives (TP) plus True Negatives (TN) to all observations, shown as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.16}$$

Accuracy describes how well the model does at predicting correct values, however, it is not always the best metric for the problem. Imagine a model which predicts cancer susceptibility and recommends tissue sampling based off of this. The model could have 90% accuracy, however, it is still difficult to evaluate this measure, because the 10% error could be either a FP, meaning the model predicted cancer and there was none, which is likely acceptable, or much more gravely, this 10% could be made up predominately of FN values, where the model predicts benign tissue and the patient has cancer - a much more deleterious scenario.

Tackling these issues are the next measurements, which are also important in the evaluation of ASAG methods in this paper for reasons which are given as the measurements are described.

Precision

Precision is the ratio of true predicted positive values to total predicted positives.

$$Precision = \frac{TP}{TP + FP} \tag{3.17}$$

Precision answers the question: of all the students where the model predicted positive i.e. *pass* how many actually passed? It is a useful measure when the FP costs are high, for instance, in spam detection identifying an email as spam when the email is benign can incur high costs i.e. the user may not be able to receive an important email. In the context of ASAG, low precision values means that the model is scoring student responses highly which in the real world should actually receive lower scores.

Since most of the current use-cases for Stembord are for low-stakes assessment, i.e. quizzes, homework items and tutorials, precision, though giving more information about a model is not as important to this paper's ASAG task as recall.

Recall

Recall is the ratio of correctly predicted positive values to all true positive values.

$$Recall = \frac{TP}{TP + FN} \tag{3.18}$$

Recall answers the question: of all the students that passed the question, how many did the model actually recognize? This is an important metric for this thesis's research approach. The models should strive for high recalls, since it can be very frustrating from a user perspective if a student knows the answer, responds to

the item correctly, and yet the model assigns a low score for that student. This situation needs to be minimized and recall is the measurement which can guide understanding of this attribute.

F1 Score

Finally, there is the F1 score which is the harmonic mean of precision and recall, which means it takes information from both into account.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3.19}$$

Because the datasets being used in this paper are imbalanced, the F1 measure, which is a stricter measure and somewhat controls for class imbalance by weighting the average - although mainly when there are many TN values - is considered a better metric than accuracy for analyzing the ASAG problem and is presented instead of accuracy in most experiment results.

3.4.2 Predicted Error Differences: Mean Absolute Error (MAE)

Returning the tuples of the form $(s_{true}, s_{predicted})$, one can also look at how far off, on average, each prediction, made by a model, is from the true value. Because the predicted scores and gold standard scores are discretized numeric values this makes sense. It is also, in general, an important piece of information when evaluating models. For instance, when predicting scores on the scale from 0 to 5, it makes a significant difference whether the model is off on average by 1 point, 0.1 points or 2 points.

The Mean Absolute Error (MAE) metric can indicate this important differences between various proposed automatic scoring solutions. The MAE iterates through a list of tuples taking the absolute value of the difference between the predicted score and the actual score. It then takes the average of all these differences. The equation is

$$MAE = \frac{1}{n} \sum_{j=1}^{n} |y_j - \hat{y}_j| \tag{3.20}$$

where n is the length of the list containing the $(s_{true}, s_{predicted})$ tuples and y_j is the j-th true score and $\hat{y}_j$ is the j-th predicted score.

3.4.3 Prediction Agreement: Cohen's Kappa and Pearson's R

Pearson's R and Cohen's Kappa are ubiquitously scattered throughout the ASAG literature, the reasons for why both of these measurements are highly touted are given in the following sections on each method.

Pearson Correlation Coefficient (PCC or Pearson's R)

Pearson's R is defined as the ratio of the covariance of two variables X and Y to the product of their standard deviations.

$$PCC = \frac{cov(X, Y)}{\sigma_X \cdot \sigma_Y} \tag{3.21}$$

It measures the linear correlation of two variables and ranges from -1 to 1, where -1 indicates that the variables are inversely correlated i.e. when the true score is 5 the model predicts 0 and when the true score is 0 the model predicts 5. Whereas a correlation of 1 means the variables are directly correlated i.e. the prediction tuples look like $(0, 0), (1, 1), (3, 3), (5, 5)$. A PCC of around zero means that the variables have no correlation with one another.

This translates to understanding an ASAG model by looking at how highly the predicted scores and actual scores are correlated, thus superior models will have PCC values which move away from 0 and approach 1, whereas poorer models will hover around zero.

Cohen's Kappa

As discussed at various points in this thesis, grades are highly subjective and the grades of any two human graders will only coincide a certain percentage of the time [Koh99]. Thus, ultimately, a comparison needs to be made between the predictions of the computer model and those of a real human judge, this is where inter-rater agreement comes into play. The core question is: given a certain level of agreement between a computer model and a human judge as it pertains to the score of a set of response items, how significant is that agreement? Further, how much does that agreement differ from what would be expected from two human judges or random chance? These are the questions that Cohen's Kappa attempts to address. It calculates a value that expresses the level of agreement between two annotators (in this case a human judge and a computer model) on a classification problem controlling for random probability of agreement [Coh60]. It is defined as

$$\kappa = (p_o - p_e)/(1 - p_e) \tag{3.22}$$

where p_o is the observed agreement ratio (the accuracy) and p_e is the expected agreement when both annotators assign labels randomly [Coh60].

In a large case-study for C-Rater, the kappa score agreement for c-rater/human on short text responses was .77 [LS04]. According to Fleiss, this represents an excellent level agreement, Fleiss further states "values greater than 0.75 or so may be taken to represent excellent agreement beyond chance, values below 0.40 or so may be taken to represent poor agreement beyond chance, and values between 0.40 and 0.75 may be taken to represent fair to good agreement beyond chance" [FLCP03]. These values roughly coincide with those of the upper bounds calculations given previously for a well-performing model.

3.5 Technical Evaluation Sections (TES) for Evaluating Experiment Results

In order compare the results of the variety of experiments carried out in this thesis, a unifying, consistent and structured way of evaluating the results is needed. In order to aid the reader in comprehending and analyzing the various outcomes, the same measurements are examined in every experiment. They are informed by the motivating questions and problem definition and attempt to provide consistent structure for both qualitative and quantitative analysis of experiment results as one journeys from experiment to experiment.

3.5.1 TES.I - Effectiveness at Automatic Grade Assignment

This section of an experiment analyzes the effectiveness of the experiment at automatically scoring the student response items. It uses charts and graphs and extracted examples to attempt to understand why a particular experiment gave its results and tries to draw conclusions and insights based on those results.

3.5.2 TES.II - Time and Space Complexity

This section looks at the experiment from both an algorithmic as well as an implementation perspective. In it there is an analysis and discussion of the algorihtmic trade-offs of the proposed approach to ASAG as well as a look at particular difficulties or lack thereof for an implementation within a LMS.

3.5.3 TES.III - Amount of Teacher Expertise and Effort Required

This section examines how much teacher effort is needed in order to carry out the proposed approach. It is also forward looking in that, in some cases, it offers ways the method could also be improved from teacher input or increased teacher training or expertise. Some experiments use methods which are fully automatic and require no teacher expertise or effort, for those this section is left out.

3.5.4 TES.IV - Fulfillment of Desired Characteristics for Stembord

This measurement includes a qualitative analysis of the degree to which the experiment's approach provides informative feedback, how well it builds or augments a useful knowledge base, as well as the method's effectiveness across classroom types within Stembord.

Summary

In general, all of the above four Technical Evaluation Sections (TES) for an experiment can occur, however, if for some reason a TES is not relevant for a particular experiment then it is left out, since there is no reason to list a section only to state that is not relevant for the given experiment.

4

Experiments

> "The first principle is that you must not fool yourself and you are the easiest person to fool."
>
> — Richard P. Feynman

The experiments as presented over the following pages, are structured in much the same way and order as they were performed during the research process, itself guided by the motivating questions presented in Chapter 2. Overall, these experiments can be classified into five categories which are outlined here and are informed by the motivating questions inherent to ASAG, arising from the constraints of Stembord and further specified by the problem definition itself. The journey begins with an exploration of experiments in the *Pattern Matching* category which introduces many of the problems in ASAG and sets a general baseline for the remaining experiments. It then progresses to an exploration of *Lexical Structure*, which tries to touch the limits of the ASAG problem space while ignoring semantics and syntax as much as possible. Following these two initial forays, experiments in *Syntax* using edit distances and the Bleu score as well as *Semantics* using WordNet and Vector Space Models are explored. The final experiment of the chapter wraps up the research for this thesis by combining the previously explored techniques with machine learning to research a hybrid approach to ASAG.

4.1 Pattern Matching

As discussed in the introductory chapter, the idea of using patterns to to find concepts in student responses is one of the oldest in ASAG. In fact, at least one LMS software product on the market, Moodle, has ASAG in the product offering [Moo18]. In the ASAG module in Moodle, teachers are allowed to create a list of regular expressions per question which are then matched against a student response. Teachers can additionally set weights for the grade a student should receive based on a particular match, for instance, a question item may have three regular expressions associated with it such as *"fuel*oxygen"* with a score of 100% and *"*fuel*"* with a score of 50% and *"*oxygen*"* with a score of 50%. Given two student responses "fuel and oxygen" and "It contains oxygen", the former would receive 100% of the points for that item whereas the later only 50% [Moo18].

Since pattern or concept matching is one of the first and oldest techniques developed in ASAG and it is a solution already on the market, it was decided to start the research with an evaluation of this technique in order to set a baseline for further evaluations.

4.1.1 Experiment 1: Teacher Created Patterns

Background, Reasoning and Hypotheses

The goal of this experiment is to evaluate the effectiveness of teacher created patterns in the form of regular expressions as a means for allowing professors to automatically grade student short answer responses. Since this method is already being used in at least one other LMS software product on the market, the results can serve to illuminate key problem areas in an ASAG approach in real-world software products. This experiment probes the problem space rounding out the lower bounds of acceptable performance just as the hand-grading task helped define the upper bounds of potential performance by giving human level correlation and accuracy scores.

This experiment uses version 1.0 of the Mohler dataset which contains 21 unique question prompts. Each of these unique question prompts was extracted and entered as rows in a CSV file which contained two additional columns for allowing test participants to enter in a list of regular expressions for matching against student responses. Participants received this file and were instructed to fill it out with regular expressions to match potential student responses to model questions. An excerpt from one of the participants completed CSV files is shown in Fig. 4.1.

This CSV file together with a short description of how to fill in the empty cells for *Full Credit Regex* and *Partial Credit Regex*, as well as a link to a tutorial on how to create regular expressions to match student responses, was sent to a software developer and a high school teacher. The author of this paper also filled out one of the CSV files, making the sample size $n = 3$. Additionally, after running all the experiments the author also attempted to squeeze as much accuracy as possible from the technique by filling out a fourth CSV file, this took many hours over the course of several days, was done by looking at student responses, and had

Prompt	Full Credit Matches	Partial Credit Matches
What does a function signature include?	{ "name_of_function" : ["names? of (function|method)s?", "(function|method)s? Name", "(function|method)'s name"], "types_of_parameters" : ["types? of(.*)(parameter|argument)s?", "(argument|parameter)s? types?"] }	{ "name_of_function" : ["names?", "functions?"], "types_of_parameters" : ["(parameter|argument)s?", "(argument|parameter)s?"] }
What is the scope of global variables?	{ "file_scope" : ["file", "files? scope", "file's scope", "scope of (file|script)", "scope of the file", "(program|script) scope", "(entire|whole) program", "anywhere in(.*)program", "whole code", "entire script"] }	{}

Fig. 4.1. Respondent CSV File: notice that in the *Full Credit Matches* column, as instructed, the respondent has created hash maps containing the name of a concept as the key, for example, *name_of_function*, and a list of patterns (regular expressions) to match for that concept as the value. Also notice how the second row does not provide any partial credit regular expressions.

purely the aim to see what the maximum performance was that this technique could achieve. This last attempt is referred to as *SE1_ext* (Software Engineer One Extended) throughout this experiment, because the author is a software engineer and the first test participant.

After the respondents returned the filled in CSV files, these were evaluated by using a function, the algorithm of which is shown in the algorithms appendix at Alg. 1. This algorithm iterates over all 630 student responses in the Mohler v1.0 dataset and scores each response by using the appropriate regular expressions for that question item in the respondents CSV file.

The maximum score obtainable in Alg. 1 is 1.0 when all regular expression concepts are matched. Conversely, the minimum possible score is 0.0 which occurs when no match can be found. If, after running all the full credit matching regular expressions, the student doesn't have a perfect score, then the partial credit regular expressions are used to check if that student can receive extra points for matching partial ideas and concepts for which the regular expressions are trying to check. For example, if a student enters *"functions have a name"* as a response to the prompt *"What does a function signature included?"*, the full credit regular expressions might search for both functions having a name as well as parameters. None of these will match. Therefore, the student's score is 0.0 after a full match search is complete. However, this also means that if the teacher has created partial credit regular expressions they will then be searched for a match using the same process.

In this example, perhaps the teacher has created a regular expression of the form *functions(.*)name* which will match and boost the student's overall score by the amount of partial credit weight the teacher has chosen. In all the experiments, the partial credit weight used is 0.5, which is exactly one-half of the full credit value.

Hypotheses

Overall, a primary goal was to assess how long it took a teacher to create regular expressions for a question item, and what accuracy could be expected from this approach under unsupervised situations i.e. when the teacher cannot view student responses but must imagine how they might potentially look. Much like the hand-grading task, this experiment is designed mainly as a baseline for evaluating later experiments. Both the author of this paper and the software developer test participant have intimate knowledge of how regular expressions work and so it was hypothesized that they would have a distinct advantage over the high school teacher and would likely have higher accuracy and correlation scores because of this.

A second and related goal was to determine what the maximum performance possible is given a supervised situation i.e. where the teacher can look at and inspect student responses. Being able to look at student responses should increase the accuracy measurements, since the teacher can directly see the answers which are correct and write patterns to match for them.

An additional expectation going into this experiment was that accuracy of this method would heavily depend on the quality and robustness of the regular expressions. For instance, if a student response contains "types of arguments" and the only regular expression is "types(.*)parameters?" then that regular expression will not match the student response. In this case, although the response is correct and should consequently receive a high score it will receive a low score or fail instead.

These ideas are formalized in the following hypotheses.

1. H_1: People who already know how regular expressions work will create regular expressions with higher precision and recall scores when matching against student response items than those who have to learn regular expressions for the first time in order to automatically grade responses.
2. H_2: Creating patterns in tandem with viewing many student responses will increase accuracy measurements.
3. H_3: Precision scores will be relatively high related to other scores because when a pattern matches it will tend to correctly identify a concept, however, due to the variety of language, these patterns should not match very often and consequently the recall and accuracy scores should be lower than precision.

Exp. 1: Presentation of Results

Each respondent self-reported that it took them approximately one hour to create the regular expressions for each of the 21 question items. This means that,

roughly speaking, three minutes per question were required to create a set of regular expressions for a given question prompt. The high school teacher reported that he required an additional hour to go through the regular expressions tutorial and learn the fundamentals of how regular expressions work. This means that for this sample and a given 5 question quiz a teacher would likely need around 15 minutes to build the regular expression pattern matchers, which seems reasonable for low-stakes tasks but is clearly an additional burden compared to solely writing model answers. Additionally, teachers unaware of how regular expressions work would need time to learn them, the teacher in this study needed one hour to learn enough in order to create matchers for 21 questions.

Exp. 1: TES.I-IV

TES.I: Automatic Grade Assignment

The accuracy measures for each of the three experiment participants are given in Fig. 4.2. The rightmost and fourth entry includes the author's hand-crafted time extensive version *SE1_ext*, for those interested in more statistical details in tabular format see Tab. A.2 in the appendix. The other important measurements, namely, the R, Kappa and MAE are given in Fig. 4.3.

Probably the most surprising result is that the high school teacher's pattern had better performance than those of one of the software engineers and were on par with those of the other. This provides direct counter evidence for the initial hypothesis that those already acquainted with regular expressions would build better regular expression matchers than those who first had to learn what regular expressions are and how they work before constructing matchers. This is significant in that it shows that after about one hour of training this high school teacher was able to build matchers on par with two software engineers well accustomed to using regular expressions in their daily work and says something about the relative ease of learning and applying regular expressions.

Far less suprising and providing no reason to reject H_2, is the fact that when the author built patterns in tandem with viewing student responses the accuract measurements increased substantially, albeit with an enormous amount of extra time and effort.

As can be seen in Fig. 4.2 recall is quite low, indicating that the regular expressions are assigning many failing scores to correct responses. This is due, in some part, to the way the regular expressions matching implementation works and is consistent with the expectations made by H_3. If a teacher only has one concept and a list of regular expressions which can match for that concept, for example, the concept *function has a name and parameters*, then either the student responses match or they do not thus the final score can only be 1 or 0 unless the teacher has also added partial credit regular expressions, in which case a score could also possibly achieve a 0.5. Note that these are only three possible scores - before scaling for evaluation - yet for any given question the human graded scores for student responses range freely from 0 to 5 over 6 classes ($0 - 5$ in steps of 1). This creates

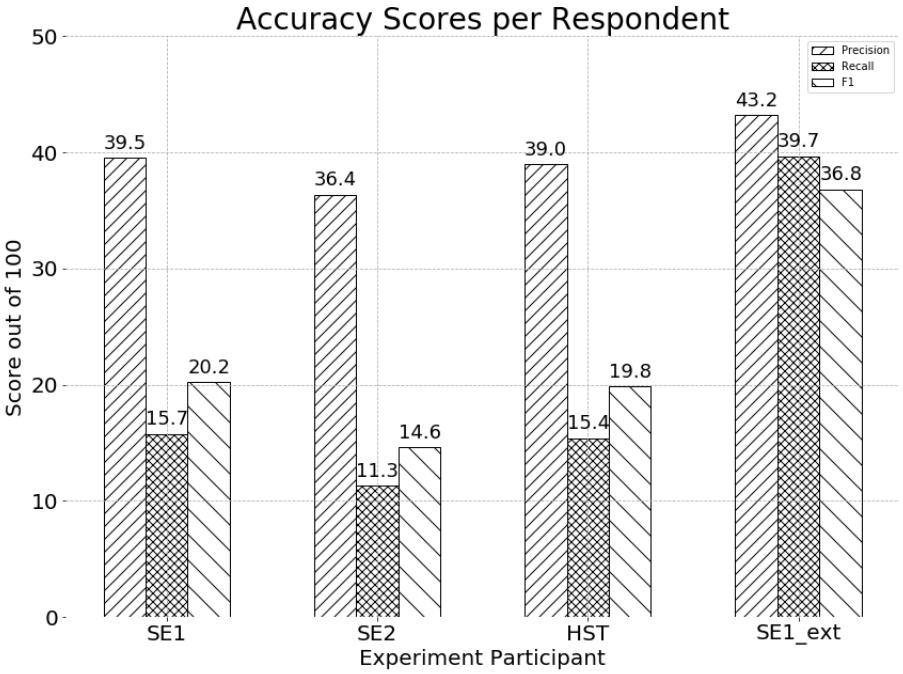

Fig. 4.2. The image shows the precision, recall, and F1 scores per respondent for 6-class classification. SE1 is the author of this paper who is a software engineer, SE2 is another software engineer, HST is a high school teacher, and SE1_ext is the version where this paper's author spent hours over the course of multiple days hand-crafting patterns by looking at student responses in order to get the best performance possible. The SE2 scores are lower than those of HST, providing evidence against H_1.

a lot of space for error when running comparisons but might not necessarily be indicative of a poor underlying method for scoring student responses in general. However, this difficulty of finely grading responses is part of any method which tries to grade based on a list of concepts on an all or nothing basis. Looking at the binary classification problem mitigates against some of these errors and is examined shortly.

As for the measurements, precision is quite high, because when an expression matches it tends to be a good indicator that the underlying concept is being expressed in the student response. This is also consistent with H_2. However, recall is quite low and upon examining the results there is one main reasons for this which is shown in Fig. 4.4. The figure shows a typical example for which the regular expression method completely missed a correct response due to its inability to generalize beyond its fixed set of patterns. In this instance, the human grader gave the response a perfect score but the automated regular expression approach missed, falsely giving a failing grade of zero.

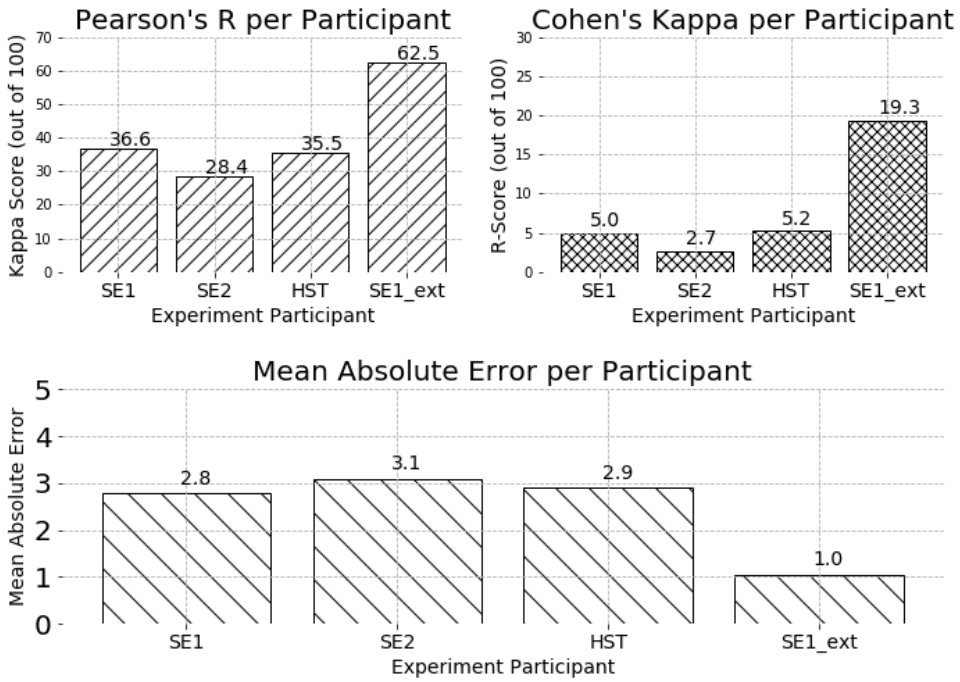

Fig. 4.3. Shows the R, Kappa, and MAE scores for each participant. Kappa shows very little association, R is around 35 for the best performers and the MAE is around 3. All of these indicate a lot of room for improvement. When comparing the best performing entry (the hand-crafted time extensive solution **SE1_ext**) to the maximums calculated after hand-grading the dataset there is still a lot of room for improvement.

Finally, this approach was evaluated for its merits at simple binary pass/fail scoring. The results for binary classification of pass/fail scoring are shown in Fig. 4.5 with additional details shown in the appendix in Tab. A.3. One can see that the best performing approach **SE1_ext** required many hours of hand-crafted regular expressions and represents the maximum achievable performance of regular expression pass / fail ability for the dataset. As in Fig. 4.3, recall is still quite low, which would most certainly lead to frustration by end-users of an ASAG system graded with this approach. For instance, in the best case hand-tweaked scenario 2 out of every 5 correct student responses are misclassified as failures. Still, when this approach does label a student response as passing it is correct almost 9 times out of 10.

Overall, these numbers provide solid baselines and an interesting perspective on a current system available on the market for ASAG. If these results were to hold up consistently given larger test subjects n then one could say that the current software is probably quite frustrating and not very effective at ASAG for its users.

TES.II: Time and Space Complexity

The algorithm shown Alg. 1 in the appendix runs in $\mathcal{O}(n^2)$ time, whereas the inner iteration, which is the call to grade a single question/response item pair

```
Prompt: How many constructors can be created for a class?
True Score: 5.0
Predicted Score : 0.0                    (A)
Model Answer: Unlimited number.
Student Answer: by default just one, but they may be overloaded
                to create as many constructors as necessary.
Full Credit Regex: {                                          (B)
    "unlimited" : [
        "infinite",
        "unlimited (number|amount)",
        "unlimited",
        "as many as you (want|need|like)",
        "as many as (are needed|needed|wanted)",
(C)     "no limit",
        "any (number|amount)",
        "as you need",
        "as many as(.*)want(.*)create"
    ]
}
```

Fig. 4.4. Example illustrating the overall brittleness of the regular expression approach due to its full reliance on human created patterns. **(A)** the correct concept should have something to do with unlimited or very many. **(B)** The student response can be inferred to create very many or infinite constructors by its wording i.e. it is correct. **(C)** There are regular expressions which are worded quite closely to the student's portion of the response that contains the concept, however, none of them actually match, therefore the algorithm falsely assigns a score of 0.

runs in $\mathcal{O}(n)$, where n is the number of patterns the teacher has defined. More precisely, this means the code for grading a single response item has a quadratic time complexity of $\mathcal{O}(n^2)$ in general terms, or $\mathcal{O}(f \cdot n + p \cdot n)$ where f is the number of full credit regular expressions, p is the number of partial credit regular expressions and n is the length of the given regular expression. From a space complexity perspective, this technique only requires the memory necessary to store the set of regular expressions and the student response and can, therefore, be considered quite efficicent and practical easily being able to run even on old hardware.

TES.III: Amount of Teacher Expertise and Effort Required

This approach requires that teachers and other not necessarily technical persons learn technical details about regular expressions and the intricacies of how they work in order to create matchers against potential student responses. The overall effort seems minimal, at least in this tiny sample - one hour to learn how regular expressions function - but it could be a deterrent and impediment to teacher's wanting to automate the process of grading themselves, since they may not have the hour or determination to devote to the task and learning an entirely new subject.

Further, this method does automate the short answer grading process to a large extent, although not completely. A teacher must spend time and effort creating a

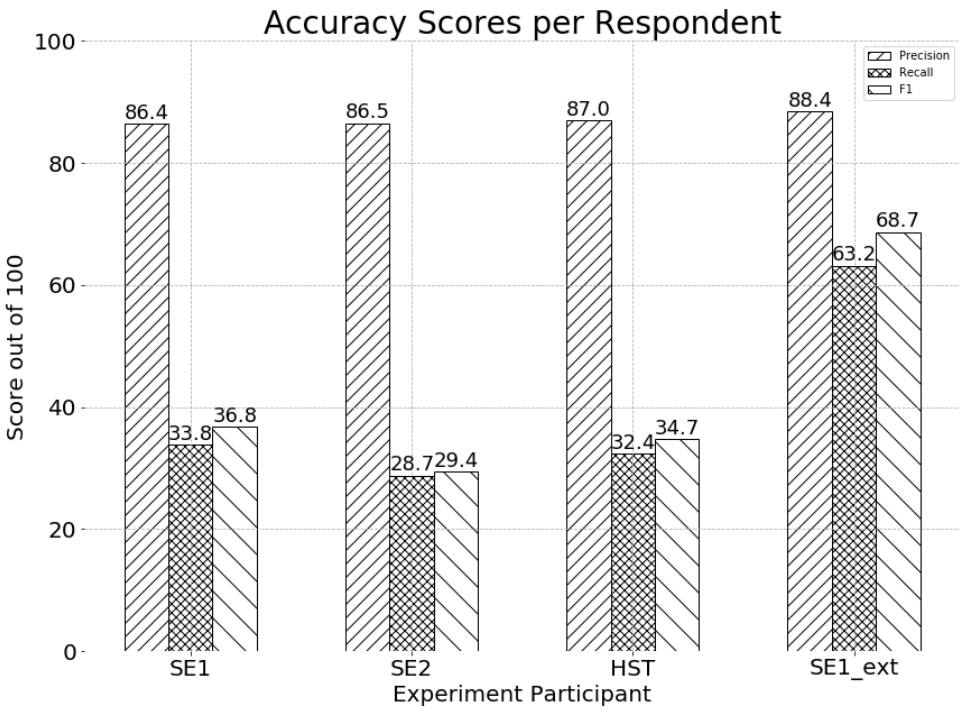

Fig. 4.5. Precision, recall, and F1 scores per respondent for the task of binary classification (pass/fail). The precision is relatively good - over 85% in all cases, however, on binary classification this method classifies roughly 7 out of 10 answers which are correct as failing. This would be extremely frustrating for students in an online course. Even the hand-crafted elite version fails on 2 out of 5 correct student responses - though when it does decide to pass a student response it is correct in doing so 9 out of 10 times.

set of regular expressions, which in the case of this study took on average about 3 minutes per question. However, once the teacher has gone through the effort to create regular expressions and assign a weight for partial credit, the teacher does not need to concern himself further with that question item. Additionally, any student responses for the question item can be automatically scored without human interaction for as long as students are taking that particular quiz.

However, it is very important to note that for the unsupervised approach the results are essentially abysmal, barely better than random chance or the lower bounds that were established in section 3.2. Only for the supervised approach, which took an order of magnitude more time (10^0 vs 10^1), and consequently provides a very negative user experience, did the results begin approaching reasonably decent performance.

TES.IV: Fulfillment of Desired Characteristics for Stembord

Formative Feedback

Providing formative feedback is probably the most shining characteristic of the pattern matching based approach. Because regular expressions match a substring within a text sequence, each match can store a reference to the start and end indices of the matching substring within the student response. This makes the approach highly transparent for both teachers and students. As seen previously and demonstrated in Fig. 4.1, teachers create a label for each concept to be matched and a list of patterns for finding student responses which contain that concept. This allows a system to automatically show the teacher and student which concepts matched and whether the concepts received partial or full credit. A hypothetical GUI rendering of a system using this method is shown in Fig. 4.6.

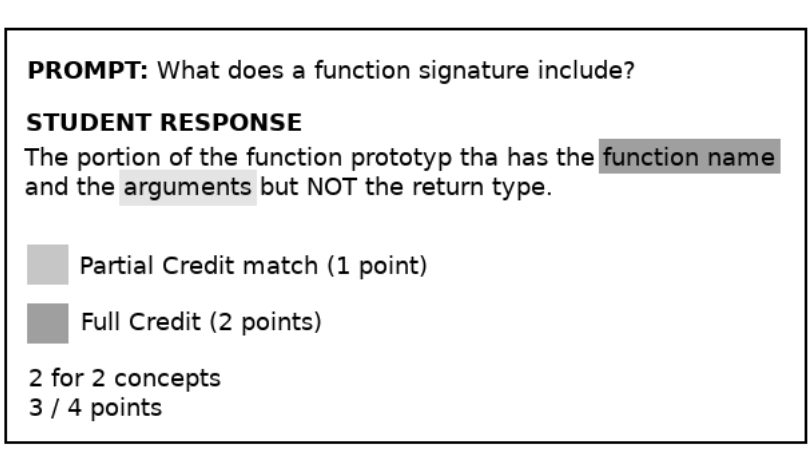

Fig. 4.6. Rendering of an automatically graded student response after feedback has been shown. A user of such a system can easily and transparently observe which concepts were detected by the algorithm, how many points were received per concept and whether a concept was a complete or partial match.

Because indices of regular expression matches within substrings can be stored, the substring which matched a particular concept can be highlighted for the student in their response. This makes the grading and scoring very transparent. This feedback feature is available in any technique which can use both concept labels and store references to substrings which match those labels in the student response.

Works for Self-Study and Teacher-Guided Classrooms

This technique would work for both *teacher-guided* and *self-study* classrooms. A teacher would create a self-study classroom and set of lessons and quizzes. In each quiz with a short answer question item the teacher would create a set of concepts and bags of regular expressions which match for text which contains those concepts, optionally adding partial credit regular expressions. After this, the teacher could

publish the classroom and any student would be automatically graded by the regular expressions when responding to one of the short answers on a quiz.

Effectiveness Across Languages

This method requires that the teacher create a new set of specific regular expressions for each language ensuring that it is, in the sense of pattern reusage, not generalizable to another language. However, given any new language and a teacher who can understand that language and regular expressions, they could create the automatic grading items themselves for the course without time delays or having to train or use machine learning models. So, in principal, this method works for any language, however, it does not generalize from one language to another, that is, an entirely new set of regular expressions must be created per language.

Conclusion

Overall, the pattern matching approach backed by regular expressions is brittle and not effective at unsupervised ASAG. The measurements in the previous experiments often bordered or dipped below the lower bounds for acceptable performance on the automatic grade assignment task. Additionally, this technique requires an initial time investment by teachers to acquire enough expertise, which although not very extensive, is still a hurdle to those users. However, the regular expression technique does have two distinct advantages. The first is that it provides the potential for detailed and transparent feedback at a concept level and, the second, is that it is also extremely efficient and simple to implement. Finally, in order for this approach to be effective, it requires a large list of student responses and constructing patterns in tandem with inspection of these responses. This means that it must be supervised, which makes this method untenable for unsupervised scoring, additionally, to achieve higher perfomance an enormous time investment on the part of teachers in pattern development making this approach not very user friendly when viewed from the holistic perspective of an entire LMS system.

4.2 Lexical Structure

After examining pattern matching as an approach for ASAG, its been established to some degree what a current ASAG module in a piece of software on the market can achieve. Now its time to begin the explorative journey of extrapolating meaning from student responses using other techniques in the research literature. This is where experiments in lexical structure come in. They explore ways in which the similarity of two sentences can be assessed without knowing the semantic contents of words and without looking at the syntacitcal relationships among those words. It can be thought of as a bare bones baseline method on top of which other more sophisticated approaches can be examined.

4.2.1 Exp. 1: Word Overlap Measures

Exp. 1: Background, Reasoning and Hypotheses

The formulas for the various word overlap measurements were expressed in detail in Ch. 3. These methods view sentences as sets of tokens and attempt to measure how many tokens two sentences share in common in order to determine their relatedness or similarity. The reason for conducting this experiment is primarily to find out how much accuracy can be obtained from a zero-knowledge approach i.e. a method which does not require any semantic or syntactic understanding and processing. An additional goal is to evaluate the effectiveness of the variously proposed word overlap techniques and to determine which ones perform the best and why. Finally, it is hoped that examining where lexical structure techniques fail could provide hints for directions to take in semantics and syntax for additional scoring improvements.

This experiment uses the filtered version 2.0 of the Mohler dataset which contains $n = 2010$ entries. The evaluation method iterates over each student response for a given model answer, performs preprocessing which includes case normalization, punctuation and stopword removal, tokenization and stemming and applies the word overlap function to automatically score the given student response and model answer pair. The word overlap function is $f(M, S)$ as explained in the problem definition outlined Ch. 2.

In the experiment results the jaccard coefficient, dice coefficient, cosine coefficient and the custom developed faucett coefficient are all run and compared to discover the best performing measurement for sentence similarity.

Hypotheses

The first hypothesis is that this approach, much like the regular expression experiment, will have a high precision. This will be due to the fact that student responses which contain most or all of the words of the model answer will tend to be correct. However, because of the fact that a perfect score is only obtainable when the student response is exactly the same as the model answer, the second hypothesis states that the predicted scores will consistently be less than the actual scores given by human graders since noise can come in from a lot of places.

Origins for noise can be words which may have the same meaning but different surface forms since this approach, for instance, views the two words *parameters* and *arguments* just as distinctly as the words *apple* and *orange*. Difficulties for this approach are expected to also arise from misspellings and different wordings between model answer and student response. Formally, these are expressed as:

1. H_1: Student responses which are correct and contain a high percentage of the words of the model answer will be correctly scored.
2. H_2: This method will perform poorly on correct responses which use different wordings than the model answer.
3. H_3: The cosine coefficient will be the best performing method at the ASAG task since other research has suggested this as well [PAP+17].

Exp. 1: Presentation of Results

Exp. 1: TES.I-IV

TES.I: Automatic Grade Assignment

The comparative accuracy measurement results of the experiments are shown in Fig. 4.7 and Fig. 4.8. These results show that the method with the highest precision is the jaccard, and the method with the highest recall is the custom faucett overlap technique. Striking a balance between those two are the dice and cosine coefficients of which the dice coefficient is minimally better. This does shed some light on H_3, namely, that there are trade-offs between each one of these techniques and if one needs precision jaccard might be a better choice, however, for the ASAG task in online MOOCs, which requires high recall scores, the custom faucett is probably a better choice at the cost of some precision. The reason for the higher recall score in the custom technique lies in the fact that it boosts lower word overlaps, effectively being more lenient for phrases which share fewer words. This also leads to lower precision, however, because some of those answers with lower word overlaps are simply wrong and the faucett coefficient incorrectly identifies them as being correct and grants them higher scores than they should otherwise have.

Turning to H_1, the item to assess is whether questions which are correct and contain a high percentage of word overlap between model answer and student response are identified and correctly scored using the word overlap methods. The hypothesis in H_1 is that this would be the case and, indeed, as the confusion matrices in Fig. 4.9 show, the majority of *(model-answer, student-response)* pairs which have over 0.75% overlap after running the overlap function either correctly predict the outcome score or get very close, predicting a score of 4.0 instead of 5.0. Therefore, there is no reason to reject H_1.

In addition, upon inspecting the misclassifications of each of the word overlap techniques for high overlap frequencies i.e. those missclassified subportions of the problem some of which are shown in Fig. 4.9, an interesting discovery was made, namely, that false negatives are a big problem with this approach. This means H_2 is not rejected. The word overlap techniques have no concept of word similarity

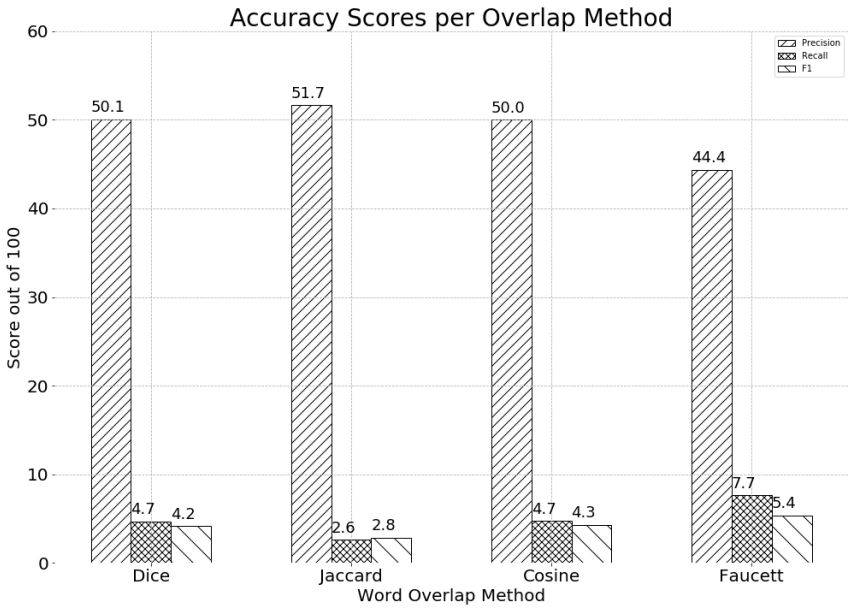

Fig. 4.7. The chart compares the scores of each word overlap technique. There are trade-offs for each method, with the jaccard coefficient having the highest precision and lowest recall scores and the faucett coefficient allowing for lower precision while increasing recall. Overall, however, the recall and F1 scores of all of these methods are far too low to be effective for ASAG.

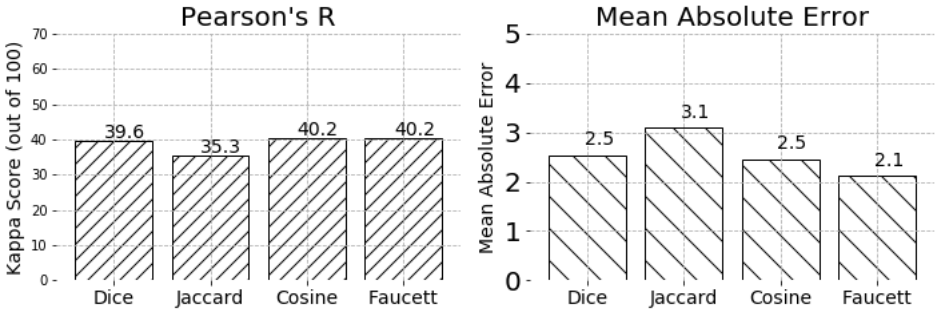

Fig. 4.8. The correlation coefficients and Mean Absolute Errors are somewhat better than the regular expression experiment, providing a hint that the number of shared tokens between two sentences does correlate with their assigned human score.

beyond surface orthographic form, and the low recall scores are indicative of this. One such example is displayed in Fig. 4.10 and shows a typical student response illustrating this situtation.

TES.II: Time and Space Complexity

From an algorithmic standpoint word overlap methods are quite efficient, but perhaps even just as important they are extremely simple to implement. First, each sentence S is converted into a list of unique tokens i.e. a set of tokens for

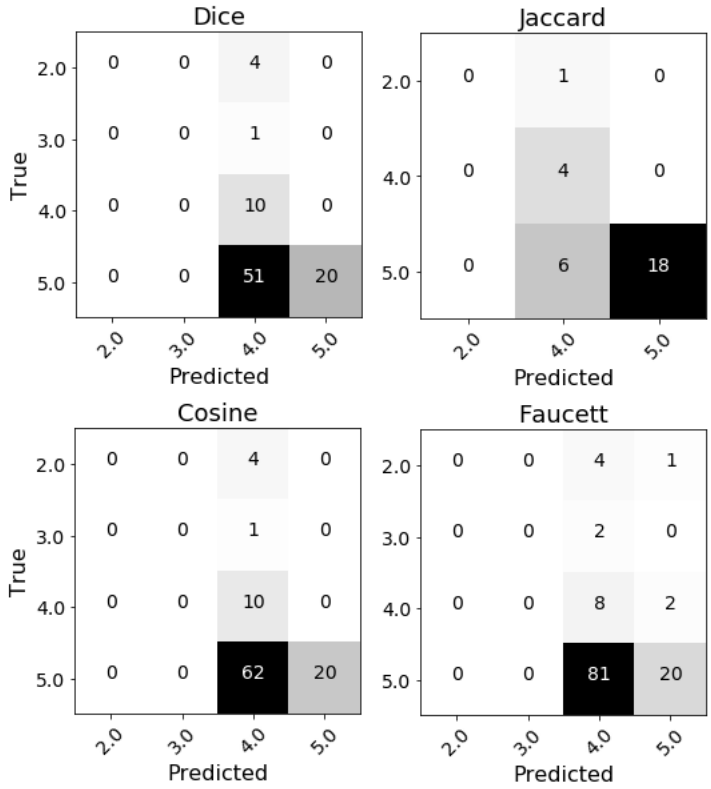

Fig. 4.9. The coefficient methods for only those *(model-answer, student-response)* pairs which have an overlap $x > 75\%$. There the methods tend to mispredict the perfect score class 5.0 as 4.0. This is due to the fact that any differences between two sentences demote a score, so even highly similar sentences will not have a perfect score unless they are identical or close to it. Still, for sentences with a high token overlap this approach seems pretty effective.

```
Prompt:             What is the main disadvantage of a
                    doubly-linked list over a basic linked list?
Score: 5.0
Pred-Score: 0.0    (A)
Model Answer:       Extra space required to store the back pointers.
Model Answer Set:   {'space', 'requir', 'pointer', 'extra',
                     'back', 'store'}
                                                                  (B)
Student Answer:     a node in a doubly linked list takes up more
                    memory than a node in a singly linked list.
Student Answer Set: {'take', 'link', 'doubli', 'list', 'memori',
                     'node', 'singl'}
```

Fig. 4.10. **(A)** the wording of the model answer and **(B)** the corresponding and correct yet different wording of the student answer. None of the tokens in the *Model* and *Student Answer Set*s are shared. This is one of the fundamental problems of the Word Overlap techniques; the inability to recognize correct but differently phrased student responses.

that sentence. This can be done in $O(n)$ where n is the length of the sentence in tokens. The methods additionally require calculating the intersection and/or union of these sets which runs in $O(n)$; this means that the performance of these methods is effectively linear i.e. $O(n)$ after removing constant operation costs.

TES.III: Amount of Teacher Expertise and Effort Required

These techniques are also simple when it comes to the amount of expertise and effort demanded of teachers, especially compared to the pattern matching experiment. A teacher needs only enter a model answer and they are done. This takes mere seconds which is much less than the 3 minute average of the pattern matching approach.

TES.IV: Fulfillment of Desired Characteristics for Stembord

Formative Feedback

Unfortunately for this approach, there is little way to provide diagnostic and informative feedback to users of the system as to why a student response received a lower or failing score. For diagnostic feedback, the only feasible approach would be to show the users the token sets of the model answer and student response and provide a statistic about the amount of overlap between the two. For formative feedback, the model answer could be shown to the student and teacher and then they would have to use their own reasoning to infer why the response received and high or low grade.

Works for Self-Study and Teacher-Guided Classrooms

The method chosen in this thesis for evaluating whether a particular technique is effective for self-study classrooms in Stembord, is to check its capabilities at all or nothing or pass/fail scoring. The results of word overlap techniques for this binary classification task are shown in Fig. 4.11.

Fig. 4.11 shows that word overlap techniques have some benefits for self-study classrooms over the pattern matching technique (when unsupervised), since word overlap techniques have much higher recall scores which consequently means less frustration for end-users. This is potentially interesting information for current implementers of software systems such as Moodle which have pattern matching implementations.

Effectiveness Across Languages

One of the shining characteristics of the word overlap approach is that it generalizes very well to any number of languages, if it was effective at the general ASAG task this would be of great benefit. Essentially, the jaccard, cosine, faucett or dice function can be programmed once and applied to any pair of sentences regardless of the language of origin and give comparable results. For those interested in the results of this approach applied to the German and Spanish versions of the dataset see Tab. A.4 in the tables appendix.

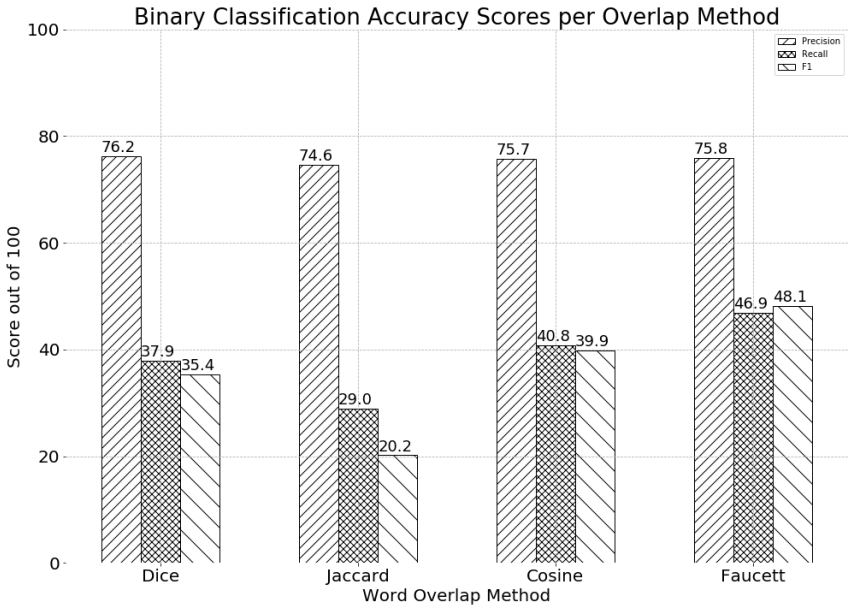

Fig. 4.11. For self-study courses which use a pass/fail scoring architecture the word overlap methods are decent, although still probably too frustrating to be usable. For instance, the best performing faucett approach fails 1 out of every 2 correct student responses and when it passes a student response that response is only correct 3 out of 4 times.

Conclusion

There are several key insights gained from the word overlap experiments. The first, is that these approaches do fairly well at scoring *(model-answer, student-response)* pairs which have a lot of shared words and are correct. They model this situation effectively except for special cases where many words are shared but the student response is still wrong. The second key insight is that these approaches utterly fail when students use synonyms or other phrasing in correct responses since this leads to low or nonextant word overlap. Fundamentally, which such low recall scores the biggest problem are false negatives for word overlap techniques, and in the context of an online LMS with a heavy focus on the user experience this is simply not acceptable because the user frustration levels would be far too high, one need only imagine answering a question correctly and being told one is wrong 3 out of 5 times to see this.

Finally, for the task of ASAG and on this dataset, details about the trade-offs between the various coefficients were determined with jaccard being the best choice for higher precision modeling and faucett the best choice for higher recalls. This precision/recall trade-off is also broadly interpreted as grading leniency for the overlap techniques. This information could be useful to other domains beyond ASAG, for instance, the more general task of short text similarity (STS).

4.2.2 Exp. 2: N-gram Comparisons

There are two other well-known lexical structure approaches in the natural language processing literature. These are using n-grams to incorporate phrasal similarities beyond simple word-to-word comparisons as well as bag-of-words (BoW) vectors which allow for counting the numbers of words or using various normalization techniques to weight the importance of certain words differently. Both are these approaches were examined and experiments were run for each, however, the results and conclusions are quite similar for both, therefore, this section only discusses the n-gram experiment since it highlights all the same points as the BoW experiment and provides some additional insights into the problem space over BoW.

Exp. 2: Background, Reasoning and Hypotheses

The n-gram comparison experiment approaches word overlap and surface structure from a slightly different angle than the coefficient equations based on set theory from the previous experiment. That word overlap experiment demonstrates that synonyms cannot be detected well at a surface level. Using n-grams should not be able to alleviate this problem. However, because n-grams take into account some minimal ordering between words - at least in consecutive sequences of two, three and four tokens - they should potentially perform better at automatic grade assignment for *(model-answer, student-response)* pairs by both taking into account some degree of word order and by looking specifically at word phrases to gain more detailed insight than just words alone in isolation. This last bit of reasoning is founded on the fact that n-grams take into account more than one word in a sequence and should, therefore, be able to to better detect common phrasing between sentences and not be as confused by student responses which simply contain some correct words.

This experiment attempts to discover which n is best when using n-gram overlap sequences for ASAG. It is an extension of the previous experiment and also explores whether having additional model answers is helpful, and if so, to what extent. The first hypothesis made for this experiment is that having more model answers should increase performance, especially, since multiple model answers could use synonyms and various other wordings and phrases which, it is hypothesized, can help alleviate the major weak point of the surface structure approaches in general. A second hypothesis states that the increase in performance from looking at similarities across n-grams will fade away as n-grams with $n > 3$ are added. This hypothesis is based off of two observations: 1) that after preprocessing removes punctuation and function words it becomes much less likely that n-grams will be shared between two sentences making overlaping sequences of $n > 3$ extremely rare and 2) other research has shown $n < 4$ to be an effective number for other related n-gram measurement tasks [PRWZ01]. Finally, it is hypothesized that using n-grams will be more effective at modeling *(model-question, student-reponse)* pairs in which word-order is important than the other word overlap approaches, simply because

n-grams store word order to some extent and do not view sentences as sets of tokens hence throwing information such as multiple word occurance away, instead they use all the information within a sentence.

These notions are formalized in the following hypotheses:

1. H_1: Having more model answers will increase the scoring accuracy of ASAG using n-grams.
2. H_2: Using n-grams for $n > 3$ will provide little or no increase in performance.
3. H_3: Using n-grams models automatically scores word order better than word overlap techniques which use sets.

The scoring function for the n-gram evaluation of a single *(model-answer, student-response)* pair is given in Eq. 4.1.

$$S(x, w) = \min\left(\sum_{i=1}^{|x|} x_i^{w_i}, 1.0\right) \tag{4.1}$$

where x is a vector of fractional values representing the ratio of n-gram overlaps between the two sentences and the i-th index reprents the size of the n-grams. For instance, x_1 is the unigram overlap ratio, x_2 the bigram overlap ratio, x_3 the trigram overlap ratio, etc. The variable w is a vector of weights with heavier weights for n-gram ratios of higher n. For instance, w_1 could be 1, w_2 perhaps 0.4 and w_3 0.2 (note that values approaching zero represent *heavier* weights since the n-gram ratios in x are all fractional values). Finally, the equation takes the minimum between the summed score and 1.0, which clips the range to $[0, 1]$ while still allowing for higher n-grams to boost the score. In the experiments the actual choice of weights was $1/n$ where n is the size of the n-gram used for the corresponding ratio value in x.

Preprocessing utilizes case normalization, stopword removal, punctuation removal and stemming.

Exp. 2: Presentation of Results

Exp. 2: TES.I-IV

TES.I: Automatic Grade Assignment

As can be seen in Fig. 4.12. n-grams are much better than single word overlap techniques at ASAG scoring, representing almost a two-fold increase over the best performing word-overlap approaches. This is also the first time that an approach is signifanctly better than the lower bounds. Fig. 4.12 also demonstrates that increasing n as hypothesized, improves grading performance, however, this increase in performance peaks at $n = 2$ which is one n smaller than hypothesized in H_2. Further, only a very small boost from $n = 2$ to $n = 3$ is provided and better score modeling flatlines showing no improvement from $n = 3$ to $n = 4$. This provides more insight to H_2, namely, that the hypothesis was wrong that n-grams of size $n > 3$ provide little benefit and instead this starts one full n-gram earlier. There is hardly a difference between $n = 2$ and $n = 3$ and one could prefer $n = 2$ over

the later for an its consequent increase in algorithm speed and performance for extremely minimal decrease in recall. This would have virtually no perceivable automatic scoring difference. H_3 is not rejected, however, as the n-gram approach models scoring better than all the word overlap methods, indicating that using all the information of a sentence and taking some small amount of order into account is a better approach for ASAG than viewing sentences as sets of tokens completely divorced of word order and word occurance counts.

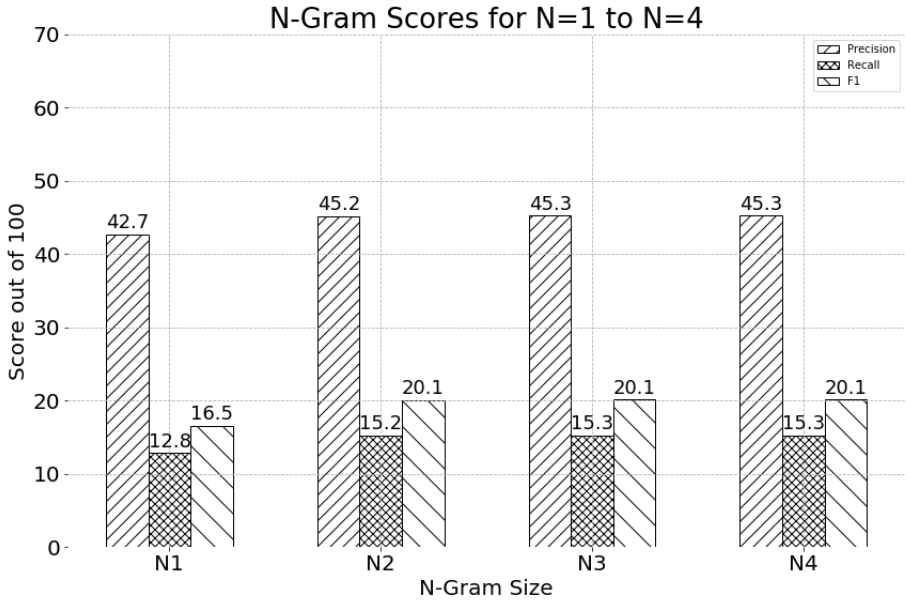

Fig. 4.12. Notice the increase in precision and recall at the transition from unigrams (N1) to bigrams (N2). Just as important is the very minimal increase from N2 to N3, which shows that trigram modeling is roughly equivalent to bigrams for this approach and perhaps doesn't warrent the increased algorithmic complexity for the tiny gain in score modeling.

Overall, not viewing sentences as sets of words but allowing for multiple word occurances and using n-gram overlaps to model the similarity between two sentences models scoring almost three fold better than unsupervised word overlap approaches, however, it still fails at sentences for which order is important and at recognizing synonyms. The previous experiment, demonstrated an example where synonym fails and although this approach performs better than word overlap since it can draw on phrases instead of just single words, it fails ust as bad at synonymy as well. This time, however, an example of word order problems is inspected to demonstrate syntactic difficulties of the lexical based approaches. This can be seen in Fig. 4.13.

Fig. 4.13 indicates that explicitly modeling the syntactical structure and ordering of words in a sentence is necessary if such questions are to be correctly graded. This leads into the subject of the next set of experiments: syntacial structure.

```
Prompt:             Order the following functions by their running time:
                    n^2; log(log n); 2^(log n); n!; n^3.
Score: 2.0
Pred-Score: 5.0
Model Answer:       log(log n); 2^(log n); n^2; n^3; n!
Student Answer:     log(log n); 2^(log n); n!; n^3; n^2
```

Fig. 4.13. The n-gram approach falsely predicts the student response as having a perfect score, whereas the human grader gave a 2.0. This is because almost all the created n-grams overlap, however, the ordering between sentences is different and significant. This information is not capture by the n-gram approach or word overlap or BoW approaches in general.

TES.II: Time and Space Complexity

Constructing the n-grams for a sentence can be done in $O(n)$ linear time, since the algorithm walks through the tokens of a sentence and stores the token and the next $n-1$ tokens in a tuple and appends this tuple to a list containing all n-grams for the sentence. Finding the overlaps between two n-gram lists is a quadratic time $O(n^2)$ operation. Therefore, the n-gram approach used in these experiments, like the other word overlap or surface structure approaches, is quite efficient, easy to implement and scalable for MOOCs and scenarios where even hundreds or thousands of students need to be graded in a matter of seconds.

TES.III: Amount of Teacher Expertise and Effort Required

This approach requires comparable time effort from teachers to that of word overlap techniques. If, however, a more robust scoring mechanism is used such as those tested in this experiment which utilize multiple model answers, then the teacher must invest the additional time in order to write and construct additional model answers which can then account for much more variety in student responses. This additional time and effort grows linearly with the number of model answers the teacher decides to write. Fig. 4.14 shows how ASAG scoring improves as the model answer count m progresses from 1 to 3; this is consistent with H_1 and due to the fact that this approach allows a student response to match against the best model answer of many, essentially increasing the acceptable phrases and constructions used in correct responses. The correlation measures which also increase with the number of model answers can be found in the appendix in Tab. A.5. Notice in Fig. 4.14 how scoring performance does not grow linearly which indicates a fast approaching trade-off equilibrium point at which the extra teacher time and effort expended is not worth the minimal boost in performance by adding a model answer.

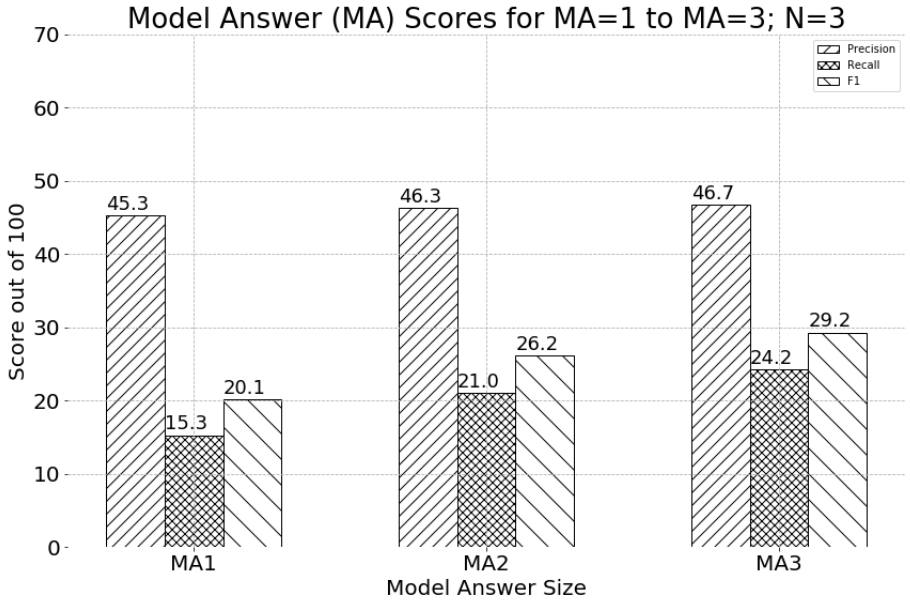

Fig. 4.14. Score modeling increases as the number of Model Answers (MA) grows from 1 to 3. Recall improves dramatically because adding more model answers allows the scoring algorithm to find correct student responses which are worded or phrased differently.

TES.IV: Fulfillment of Desired Characteristics for Stembord

Formative Feedback

Unfortunately, like the the other word overlap approaches, the n-gram method does not lend itself well to informative feedback. When multiple model answers are employed a feedback mechanism could show teachers which model answer a particular student response matched with. For students, the matching model answer as well as a possible alternative could be shown in order to let the student see other potential correct approaches to the problem. However, a detailed informative feedback mechanism like that seen in the pattern matching approach is not feasible.

Works for Self-Study and Teacher-Guided Classrooms

Fig. 4.15 shows the confusion matrix for pass/fail scoring of the best-performing n-gram model (MA = 3, N=3). It has a 86% precision for passing scores which means when it predicts a student response as being correct, the model is correct almost 9 times out of 10. Where the model performs poorly is in precisely identifying incorrect scores, there its precision is only 38% indicating the in 5 out of 8 student responses which it classifies as failing, the student response should have actually received a passing grade. This is probably not acceptable for self-study courses and would lead to high levels of frustration among users.

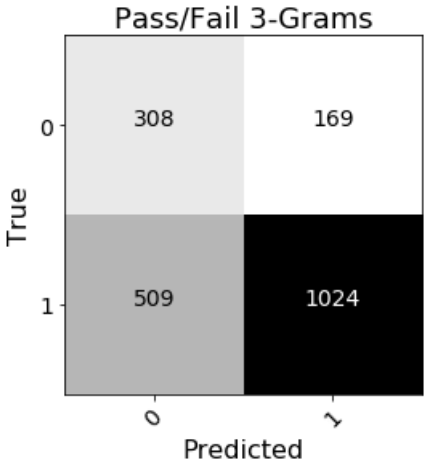

Fig. 4.15. Notice how many correct (1) student responses are predicted to fail and scored with a 0 by the n-gram model, this would lead to high end-user frustration making it difficult to apply this approach for self-study classrooms.

Effectiveness Across Languages

The n-gram approach performed comparatively across both the Spanish and German language datasets although both performed worse than their English counterpart and German somewhat more so. The details can be see in appendix in Tab. A.6. Overall, however, this indicates that a n-gram based scoring approach could easily and effortlessly scale to cover at least all languages which use alphabets, although some considerations would likely need to be made for left to right scripts such as Arabic and a new experiment would need to be made in order to assess whether n-grams are effective at all for non-alphabetic scripts such as Chinese or hybrid scripts such as Japanese.

Conclusion

Several aspects of the ASAG problem were discovered in the n-gram experiment. First, bigrams perform, for all practical purposes, just as well as $n = 3$ and higher n-grams for scoring short answers and they mark a significant improvement over word overlap approaches, this is likely due to taking all tokens in the sentences into account as well as common phrase patterns and simple word ordering between model and student answers. Additionally, incrementing the model answer count provides increased ability of n-gram approaches to score short answer responses, and this is most likely generalizable to other methods beyond n-grams as well.

There are, however, still many ways in which n-grams are lacking. First, they do not take enough syntax and word order into account to correctly grade short answer responses which necessitate understanding of ordering. Second, n-grams still have no concept of meaning or synonymy beyond surface orthographic structure, which is one reason they probably showed so much improvement when using additional model answers. The first of these two problems, namely syntax, is tackled in the next set of experiments.

4.3 Syntactical Structure

The previous experiments left two directions open which need more exploration: syntax and semantics. This section explores the former of these two. It is also unique amoung the experiments in that it does not attempt to score short answers but only tries to find correlations between syntactical measurements of *(model-answer, student-response)* pairs and scoring values. This is because assigning a grade based on syntactic similarity alone makes very little sense in the context of ASAG. To clarify this assertion, consider Fig. 4.16 which shows a hypothetical nonsensical student response and demonstrates how the meaning of two sentences can be completely divorced even though both sentences share the exact same syntactic structure.

```
Prompt:                 What is a recursive function?
Model Answer:           a function that calls itself
Model Answer POS:       [det, noun, adj, verb, pronoun]
Student Response:       a car that wrecks itself
Student Resonse POS:    [det, noun, adj, verb, pronoun]
```

Fig. 4.16. Observe how the highlighted POS (Part-of-Speech) tags are not only identical but also share the exact same order in the *(model-answer, student-response)* pair, despite the fact that both sentences represent completely unrelated ideas. This demonstrates the lack of usefulness of running experiments which use syntax alone to grade student responses.

4.3.1 Exp. 1: Distances: Edit, Tree, Bleu-Score

Exp. 1: Background, Reasoning and Hypotheses

These experiments compare the edit distance for sentences parsed into lemmas, POS tags and dependency parse tags. The tree edit distance is used as well the bleu score. These represent five distinct ways to extract syntactic meaning and relationships from *(model-answer, student-response)* pairs. The structure of the experiment is similar to previous ones, preprocessing is done for punctuation and stopword removal and whitespace and case normalization. This time, however, instead of stemming, lemmatization using the spacy library is applied since it can more effectively normalize tokens into their root forms than stemming in many cases. After this step, each of the measurements are calculated between model answer and student response for all the $n = 2010$ items in the dataset.

Hypotheses

Due to the fact that the bleu score uses n-grams and based on the previous n-gram experiments, it is hypothesized that the bleu score will have a positive correlation

to the target variable. This is due to the implementation details of the blue score [PRWZ01]. The other methods are much more ambiguous, however, it is clear that as the costs increase for the edit distance set of functions so too do the syntactical differences between model and student answers. The second hypothesis is that syntactical difference should correlate negatively with score for these methods, since two sentences which are syntactically very different are likely to be semantically different as well, at least that is the line of thought. These are formalized as follows:

1. H_1: The bleu score will correlate positively with the target variable.
2. H_2: The edit distance scores, including the tree edit distance, will correlate negatively with the target variable.

The strength of these correlations is entirely unknown and thus determining these strengths, directions and acquiring a deeper understanding of syntax measurements is the main purpose of this experiment.

Exp. 1: Presentation of Results

Exp. 1: TES.I-IV

TES.I: Automatic Grade Assignment

The results for the bleu score are shown in Fig. 4.17 which shows the histogram distribution of the blue score calculated on the entire dataset plotted against gold standard scores. Fig. 4.17 also shows the same plot but only for a subportion of the dataset made up low scored student responses. It indicates that student reponses which receive lower scores are syntactically less similar to the model answers than student responses with higher scores even though there is still a lot of noise and overlap.

Overall Fig. 4.17 shows that the bleu score could be helpful in modeling some parts of the syntactic structure, however, as the blue score decreases so does the correlation with scoring, which means that highly divergent sentences are not differentiated well using this method. H_1 is not rejected, however, since the bleu score does positively correlate with the target variable.

The correlation scores for the set of edit distance functions are shown in Fig. 4.18. The confusion matrix indicates that higher edit distance scores for all the measures correlate with lower human assigned grades. This provides no evidence to reject H_2 and is consistent with the claim that student responses which syntactically diverge more from the model answers tend to receive lower scores.

In the dataset, question 12.3 deals explicity with word ordering making it an ideal candidate to inspect for determining how well and to what degree these measurements function. Therefore, the token and tree edit distances in the context word order importance for grading is shown in Fig. 4.19 by using hand-picked examples out of question 12.3 to demonstrate the effectiveness of these methods. This figure demonstrates how higher edit distance measurements are indicative of poorer human assigned grades and how this relates to the fact that student responses altough containing the correct words have the words in the wrong order.

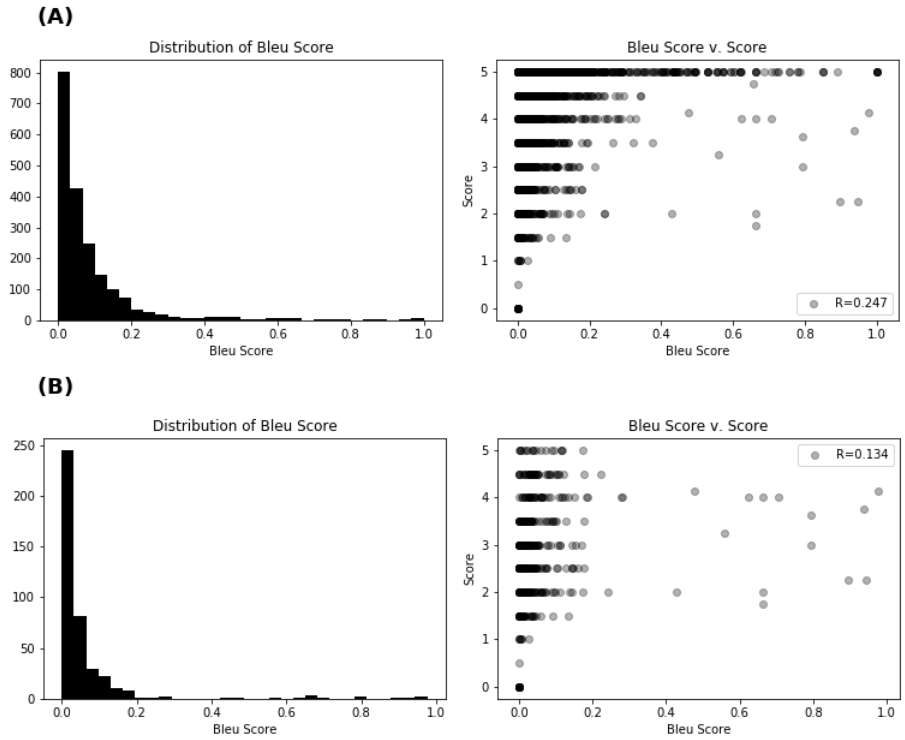

Fig. 4.17. (A) the histogram and scatter plot with correlation coefficient $r = 0.247$ between the bleu score and target variable for the entire dataset. **(B)** same as **(A)** except only for the lower scores of the dataset i.e. those with failing grades or very low scores with $r = 0.134$. The bleu score tends to be lower for lower scores. This indicates that the structure of lower scored student responses does indeed differ from those responses which receive higher grades.

TES.II: Time and Space Complexity

Computing the bleu score has essentially the same time and space characteristics of the n-gram approach and therefore runs in quadratic $O(n^2)$ time. Each of the other methods vary significantly. The token-based Edit Distance can run in $O(n{\cdot}m)$ where n and m are the two input sentences. The Tree Edit Distance is a dynamic algorithm and, using the Zhang and Shasha implementation runs in the worst case in $O(|T_1|^2|T_2|^2)$ [Bil05]. The other approaches require running a POS tagger or dependency parser over the sentence initially which has a run time dependent on underlying implementation of the POS tagger or dependency parser. The edit distance portion run afterwards has a run-time complexity of $O(n \cdot m)$ just like the token-based approach. Overall, these methods are somewhat computationally expensive but still solveable in low polynomial time and hence effective and scalable for online grading systems.

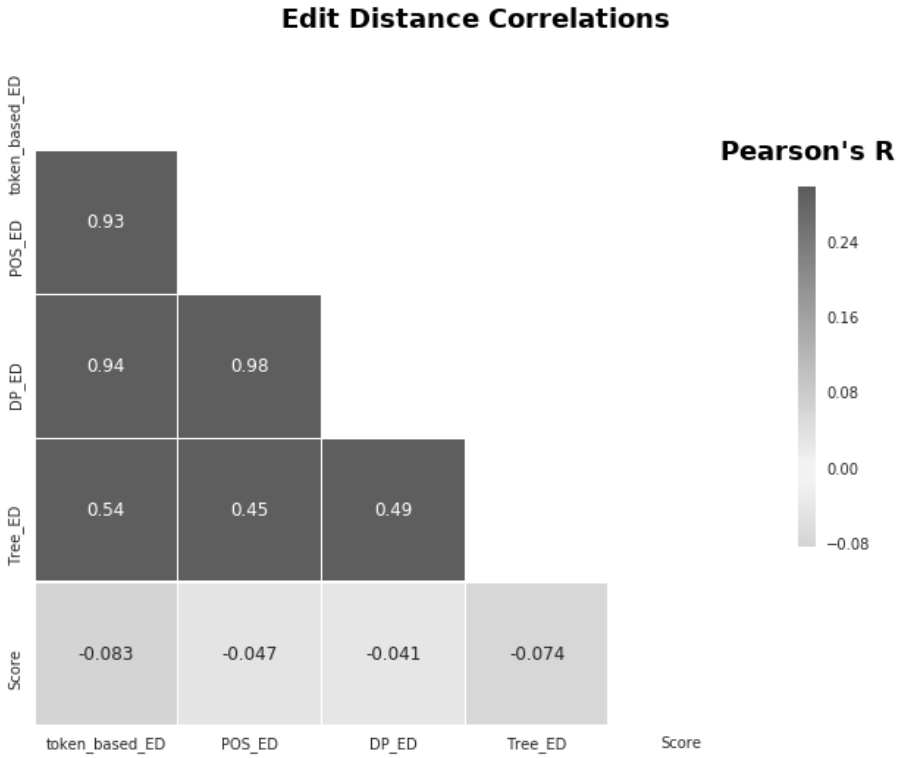

Fig. 4.18. The token based Edit Distance (token_based_ED) and Tree Edit Distance (Tree_ED) provide the highest negative correlations with scoring. The token based and tree edit distances correlate the least with one another, indicating that they are exctracting different pieces of information from the underlying student responses. These two are likely good potential feature candidates for features extracting syntactic information in a machine learning model.

TES.III: Amount of Teacher Expertise and Effort Required

Extracting syntactic information requires no additional effort or expertise from educators other than the input of a single model answer. Therefore, in terms of teacher effort, the syntactical features examined here have zero costs.

TES.IV: Fulfillment of Desired Characteristics for Stembord

Effectiveness Across Languages

The methods here were not tested against other languages, however, the techniques themselves are capable of being applied to any language. Overall, there is no reason to expect that the effects seen here such as greater synactic divergence between model and student responses correlating with lower scores should be any different across languages.

```
Prompt:             Order the following functions by their running time:
                    n^2; log(log n); 2^(log n); n!; n^3.

Example 1:
Score: 5.0
Model Answer:       log(log n); 2^(log n); n^2; n^3; n!
Student Answer:     log(log n); 2^(log n); n^2; n^3; n!          (A)
Token-Based ED:     0.0
Tree ED:            0.0

Example 2:
Score: 2.0
Model Answer:       log(log n); 2^(log n); n^2; n^3; n!
Student Answer:     log(log(n)), n^2, n^3, 2^log(n), n!;         (B)
Token-Based ED:     0.08
Tree ED:            0.06

Example 3:
Score: 1.5
Model Answer:       log(log n); 2^(log n); n^2; n^3; n!                (C)
Student Answer:     linear, logarithmic, exponential, linear, linear
Token_Based_ED:     0.17
Tree ED:            0.17
```

Fig. 4.19. Hand picked examples from question 12.3 from the Mohler v2.0 dataset. **(A)** perfect score and perfect syntactical match between student response and model answer; the Edit Distances (ED) are both 0.0. **(B)** The ordering of the student answer differs slightly from the model answer, which forces the edit distances to increase in order to transform the student response into the model answer and this transformation cost is indicative of a lower human assigned grade, here 2.0. *(C)* An example where the student response is utterly different from the model answer and consequently has the highest Token-Based and Tree Edit Distances.

Conclusion

The main insights gained are that the bleu score can identify the structure of correct answers which are syntactically similar to model answers and that the Tree Edit Distance and Token-Based Edit Distances are more effective than POS and Dependency Parse Edit Distances at extracting information about the syntax structure and relationship between model answer and student response. This is useful information for feature engineering.

4.4 Semantics

Having examined and discovered features for extrapolating syntactical relationships in the previous experiment, its now time to turn to the next problem area - semantics. As discussed in Ch. 3, there are, generally speaking, two ways of modeling synonymy and these are knowledge and corpus based approaches and distributed vector representations. The ASAG literature is full of many explorations in the former and that is where the experiments in semantics begin.

4.4.1 Exp. 1: Knowledge and Corpus Based Meaning

Exp. 1: Background, Reasoning and Hypotheses

This experiment represents the first exploration of meaning beyond the surface orthographic structure of tokens. Its goal is to determine which knowledge and corpus-based semantic measurements, using *WordNet*, are most effective at capturing the meaning relationships between words in different *(model-answer, student-response)* pairs. In order to achieve this, this experiment compares knowledge and corpus-based similarity techniques outlined in Ch. 3, subsection 3.3.5.

Importantly, the only other language besides English for which *WordNet* is available, at least in this research, is Spanish. This means the knowledge and corpus-based approaches will not be extendable to other languages on the Stembord platform besides these two. Because of this, a legitimate question would be: why even attempt to assess these methods? The answer is three-fold. First, Stembord is currently only available in English and will remain so into the near future. Further, since its target market is the USA the next language to add is Spanish. Second, if these methods prove highly effective they could serve as an interim solution until more generalizable methods can be found. Additionally, obtaining insights from knowledge and corpus-based methods should help evaluate the effectiveness of other semantic approaches. Finally, it could be the case that the knowledge and corpus-based methods tested in this experiment model meaning in subtly different ways than other approaches and therefore provide additional context and information which could serve as a feature to boost the performance for the English and Spanish language versions of the ASAG module. Although this performance boost would not be available for other languages, it would still provide a better user experience for the main target audience for Stembord, at least currently and into the near future.

In terms of the setup, the Brown corpus[1] is used to calculate the Information Content for those measurements which require it (Resnik, Jiang & Conrath, Lin). The preprocessing pipeline removes stopwords and punctuation, carries out whitespace and case normalization and performs POS-tagging which removes all tokens except nouns, adjectives and verbs. This last preprocessing step, is a prerequisite only because of the way *WordNet* is structured, i.e. *WordNet* only contains

[1] The Brown corpus is one of the datasets used by NLTK and can be downloaded at `https://www.nltk.org/nltk_data/`

detailed graph hierarchy information on nouns and verbs and to a lesser extent adjectives and adverbs. Adverbs were left out since adding them had little to a slightly negative impact on the overall scores when assessed in preliminary experiments.

Hypotheses

Based on the literature comparing knowledge and corpus-based measurements for the ASAG task, specifically Mohler 2009, it is hypothesized that the Jiang and Conrath extension of the Resnik similarity will perform the best on the dataset [MM09]. However, this experiment runs the measurements made by Mohler 2009 against a new and extended dataset, as well as one that has been pruned of easy to automatically score short answer responses, so the results could prove different when run over this different data. The second hypothesis is that adding up to three model answers will significantly increase the accuracy of the scoring methods. This is based off of previous results seen in the n-gram comparisons experiment. There, using three model answers instead of one increased the overall accuracy by almost 10%. Initially, there is no reason to expect anything different in this experiment, therefore, H_2 states this expectation. Formally, the hypotheses are:

1. H_1: The Jiang and Conrath similarity will be the best similarity measure followed by the Shortest Path, Lin, and Wu and Palmer similarity scores.
2. H_2: Using 3 model answers instead of 1 will increase model accuracy by nearly 10%.
3. H_3: Correct student responses which use synonyms and are phrased differently than the model answer will be detected, more so than in lexical or n-gram techniques.

Exp. 1: Presentation of Results

Exp. 1: TES.I-IV

TES.I: Automatic Grade Assignment

Fig. 4.20 shows the accuracy scores for the various measurements. Interestingly, the Wu and Palmer (WP) similarity score performed the best of the bunch. However, the measure with the highest correlation coefficient is the Jiang and Conrath, as shown in Tab. 4.1, followed closely by the shortest path which is consistent with H_1.

These results are fascinating because they give some deeper insight beyond what Mohler (2009) describes. For instance, the scores in Fig. 4.20 and correlation coefficients in Tab. 4.1 indicate that of the class of techniques which only make use of *WordNet* and ignore statistical corpora, Wu and Palmer (WP) performs the best, sacrificing some precision compared to the Shortest Path (SP) in order to not falsely give a student response a lower score than it deserves. This makes sense as well, because the WP method uses a common concept (Least Common Subsumer)

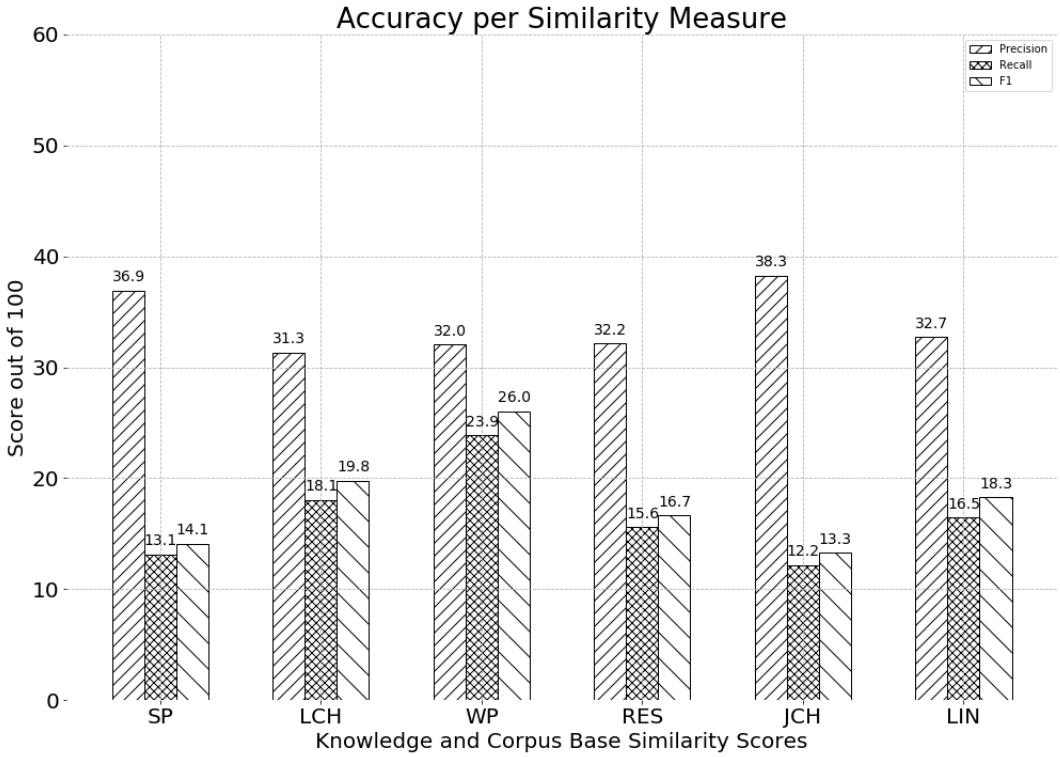

Fig. 4.20. Accuracy Scores for Shortest Path (SP), Leacock & Chodorow (LCH), Wu & Palmer (WP), Resnik (RES), Jiang & Conrath (JCH) and Lin (LIN). The Lin technique, which uses a combined approach of both the *WordNet* knowledge base and Information Content from a text corpus, was more lenient than JCH at the cost of some precision. Overall, the WP measure provides the best balance between precision and recall for the ASAG task on this dataset.

Measure	**R**
Shortest Path	0.28
Leacock & Chodorow	0.06
Wu & Palmer	0.13
Resnik	0.16
Jiang & Conrath	0.32
Lin	0.13

Table 4.1. Correlation coefficients for each knowledge and corpus-based similarity measure.

shared by two subconcepts which makes the measurement slightly fuzzy compared to the shortest path technique which uses node distances between concepts only.

Of the next class of methods which use both *WordNet* and statistical corpora and Information Content (IC), Jiang & Conrath (JCH) has by far the highest precision and correlation coefficient. Its lower recall scores are indicative of the cost of increased ability to score a student response closer to the gold standard. Since both the LIN and JCH measurements attempt to decrease the coarseness of the Resnik measurement, it appears that on this dataset and for the application

of ASAG, JCH is more effective at increasing precision at a cost of recall, whereas Lin augments all scores minimally over the Resnik.

All of the measures have a high correlation among themselves, indicating unsurprisingly that each is extracting similar information from the student responses. However, the lowest correlations are between WP and JCH which are in many ways the best methods from their respective measurement categories. This means these two are likely the best candidates for features in a machine learning pipeline.

H_2 states that a 10% gain should be seen by using three model answers instead of one. This hypothesis is drawn from the lexical structure experiments which showed almost 10% gains by using three model answers. The results are displayed in Fig. 4.21 which provides enough information to reject the hypothesis. Gains were seen, however, these are in the range of 4 to 8 percentage points, depend on the measurement, and improve recall scores much more than precision. The latter makes sense because having multiple model answers provides broader coverage of possible correct response formulations which, in turn, allows models to better match correct student responses than when there is only one source of truth.

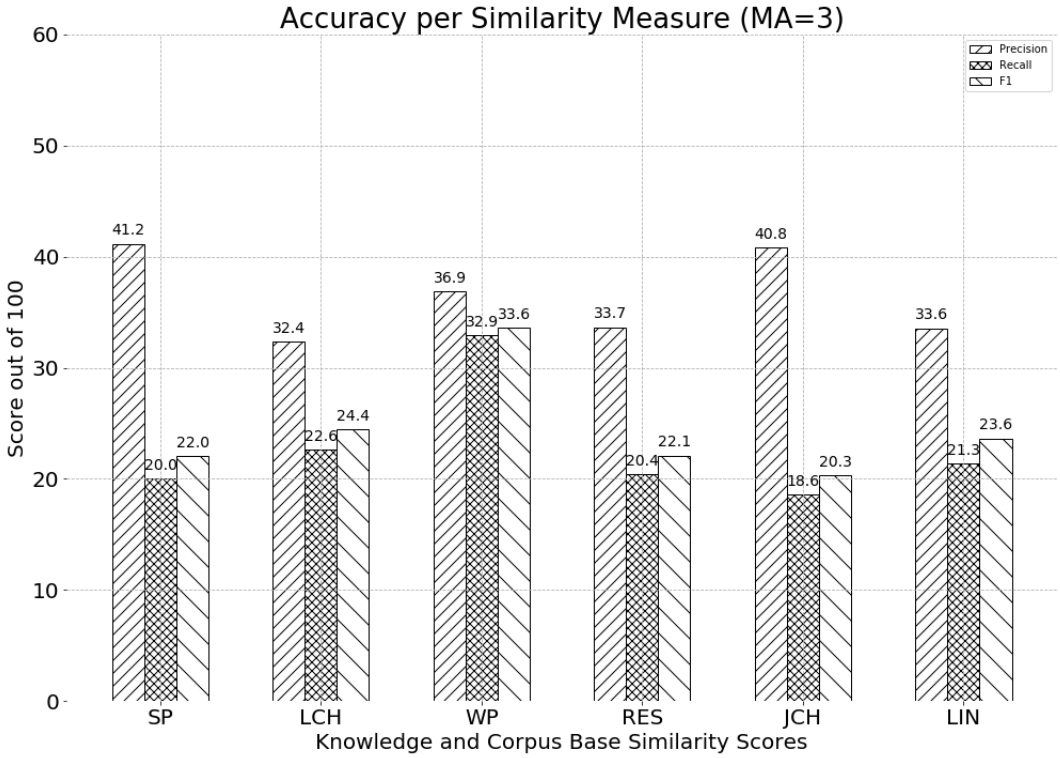

Fig. 4.21. Accuracy Scores for Shortest Path (SP), Leacock & Chodorow (LCH), Wu & Palmer (WP), Resnik (RES), Jiang & Conrath (JCH) and Lin (LIN). Overall, a boost is obtained by using three Model Answers (MA=3), however, it was less than the hypothesized 10% and had a positive impact much more on recall than on precision

Finally, upon inspecting some the scored results, it is apparent that the higher scores are due to these methods being able to pick up more detail from synonyms than approaches examined up to this point, providing no reason to reject H_3. Fig. 4.22 demonstrates a situation, which receives a higher score using this approach, precisely because synonyms and related words are being taken into account.

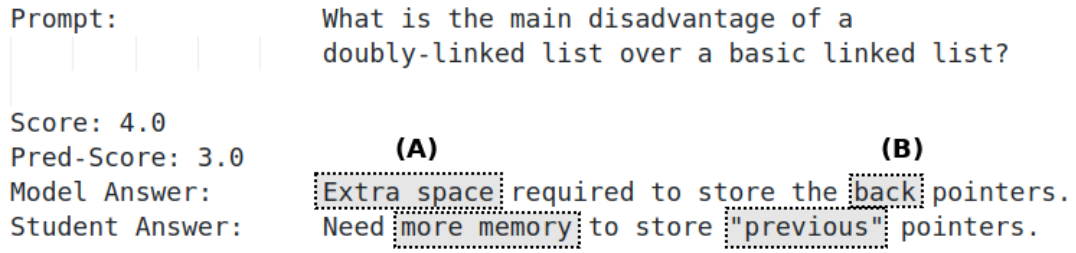

Fig. 4.22. Example of a predicted score from the dataset using the Jiang and Conrath method. **(A)** shows that the phrases *Extra space* and *more memory* are detected as related as well as the words *back* and *previous*, which all then contribute to the predicted score.

TES.II: Time and Space Complexity

These methods are computationally more complex than other techniques previously explored. Each word in the model answer m must be compared with every word in the student response n which is an $O(m \cot n)$ complexity. However, it doesn't stop there. For each word-pair formed in this process, a synset for each word must be found in *WordNet* and then for every sense in each synset the two senses which are most similar are taken and their similarity score is calculated. This results in the algorithm running in the higher polynomial time of $O(n^4)$, which means it is slower than other methods seen up to now, although it is still a polynomial time algorithm and useable for automatic scoring of thousands of student responses and, consequently, practical.

TES.III: Amount of Teacher Expertise and Effort Required

This approach requires only that a teacher enter in one or more model answers, no knowledge expertise is required beyond the subject matter of the course the teacher is creating. Therefore, this approach to extracting meaning can be considered very simple from a teacher's perspective.

TES.IV: Fulfillment of Desired Characteristics for Stembord

Formative Feedback

This method does not provide an effective means of scoring feedback other than the ability to save a map of highest similarity word correlations between student

response and model answer. This could then be color-coded to provide teachers insight into how a particular student response received its automatic score, however, this would require some training on the part of teachers and students to understand the maps and it is further not apparent that this information is helpful beyond providing more transparency into the inner workings of the scoring algorithm.

Works for Self-Study and Teacher-Guided Classrooms

Fig. 4.23 shows the results of the experiments run for the pass or fail automatic grading.

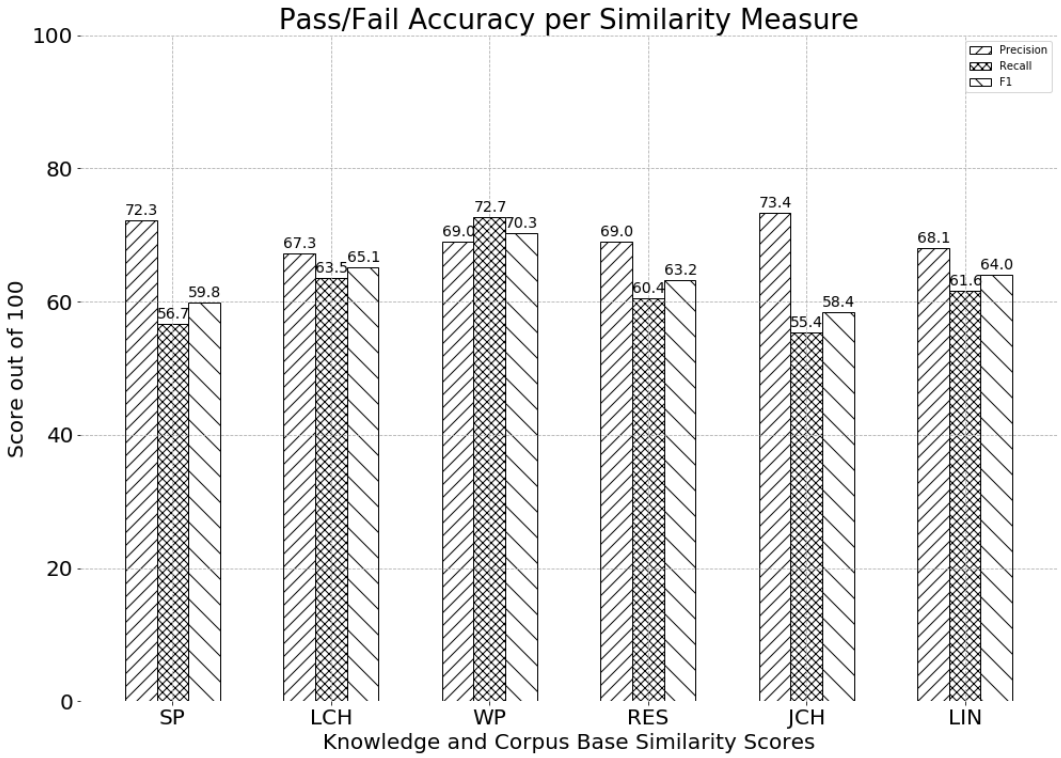

Fig. 4.23. Pass/Fail Accuracy Scores for Shortest Path (SP), Leacock & Chodorow (LCH), Wu & Palmer (WP), Resnik (RES), Jiang & Conrath (JCH) and Lin (LIN).

For self-study courses, using the WP measure would effectively mean that 7 times out of 10 when a student response is correct that it also receives a passing grade. It also means that 7 times out of 10 when a student response is incorrect that the automatic scorer assigns it a failing grade. This alone is significantly better than the regular expression approach and a large step forwards in terms of the user experience, although still probably somewhat frustrating from a user perspective when implemented in an online MOOC due to the false negatives.

Effectiveness Across Languages

As discussed in the introduction to this experiment, these techniques are largely not applicable to other languages beyond English and Spanish. However, because

NLTK comes with a Spanish version of *WordNet*, the tests were run for all measurements which do not require Information Content, that is the SP, LCH, and WP measurements. The results are abysmal and shown in Tab .A.7 in the appendix. They indicate how vitally important the extent, size, and professional quality of human created knowledge bases are when it comes to applying these techniques to identify semantic content. It also shows that for practical purposes a Spanish language version is not feasible.

Conclusion

The bottom line is that this experiment shows that the Wu & Palmer and Jiang & Conrath measurements are most different from one another and most effective at extracting meaning from words in sentences on this dataset. These methods are not broadly applicable to languages outside of English, however, they can be used to automatically grade an English language ASAG module. Finally, as seen in previous experiments, using three model answers instead of one provided a significant $4-8\%$ boost in precision and recall scores.

4.4.2 Exp. 2: Word Embeddings

Exp. 2: Background, Reasoning and Hypotheses

In the previous experiment, it was observed that analyzing the semantics of words using knowledge and corpus-based approaches can provide significant performance gains over simple orthographic approaches in which only the surface form of the word is taken into account. However, there are several disadvantages of using knowledge bases, primarly the fact that they depend heavily on the quality and size of the human created knowledge base. This was demonstrated by the poor performance of Spanish *WordNet* using the same methods as the English *WordNet* in the previous experiment. Therefore, this set of experiments analyzes an approach using word embeddings for the task of automatic short answer grading. The goal is to compare two different word embedding models *Word2Vec* and *FastText* to determine which performs best at capturing the underlying semantics of words. The experiments look at the effects of n-gram sizes, number of model answers, length penalization. They also make model comparison across languages. A final objective of these experiments is to determine how well word embeddings perform compared to knowledge-based approaches and what, if any, significant differences or contributions they provide to gaining semantic information from *(model-answer, student-response)* pairs.

The preprocessing techniques performed are whitespace and case normalization, punctuation and stopword removal and lemmatization.

Hypotheses

There are five hypotheses each related to n-grams, word embedding model, length penalization, model answers, and cross language modeling capability and are derived from previous experiments. Formally, they are:

1. H_1: Increasing n-grams will increase accuracy measures and correlation up to $n = 3$.
2. H_2: *FastText* models for learning word embeddings will capture more meaning than *Word2Vec* because *FastText* takes into account word morphology.
3. H_3: Increasing model answers will increase accuracy measures and correlation with diminishing returns after $n > 1$.
4. H_4: Penalizing student responses for their length will result in increased precision and lower recall scores since student responses which more closely resemble length and word semantics of the model answer will be better detected, however, this will be at the cost of failing to assign good scores to longer more elaborate student responses.
5. H_5: Word embedding models will have better measures than knowledge-based approaches across languages since they are not dependent on manual human construction like knowledge-based approaches.

An open question is whether word embeddings will perform better than knowledge based approaches at the task of ASAG.

Exp. 2: Presentation of Results

Exp. 2: TES.I-IV

TES.I: Automatic Grade Assignment

The answer the H_1 and H_2 as well as the open question of word embedding performance compared to knowledge-based approaches is shown in Fig. 4.24. This clearly demonstrates that *FastText* models outperform *Word2Vec* models on the task of ASAG, which is likely due to the fact that the *FastText* method takes word morphology into account. This finding represents an extension to research by Adams et al. which compared only *Word2Vec* when using word embeddings for ASAG [ARK16]. It has been shown here that *FastText* performs better at the ASAG task compared to *Word2Vec*.

Previous experiments in this thesis have shown that increasing the number of model answers significantly increases the performance of a given technique at ASAG, however, with diminishing returns after $n > 1$ for each new model answer added. H_3 explicity stated an expectation that this behavior would continue for word embeddings and Fig. 4.25 demonstrates that there is no evidence to reject H_3.

Both of these experiment results provide evidence that word embeddings perform better than knowledge and corpus-based approaches at modeling semantics and using semantic understanding for the task of ASAG. This provides an answer to the open question posed at the outset of the experiment, namely, it is the case that word embeddings do outperform knowledge-based approaches for the ASAG task.

For length penalization, the experiment results overwhelming demonstrate that H_4 should be rejected. Penalizing student responses for containing excess words

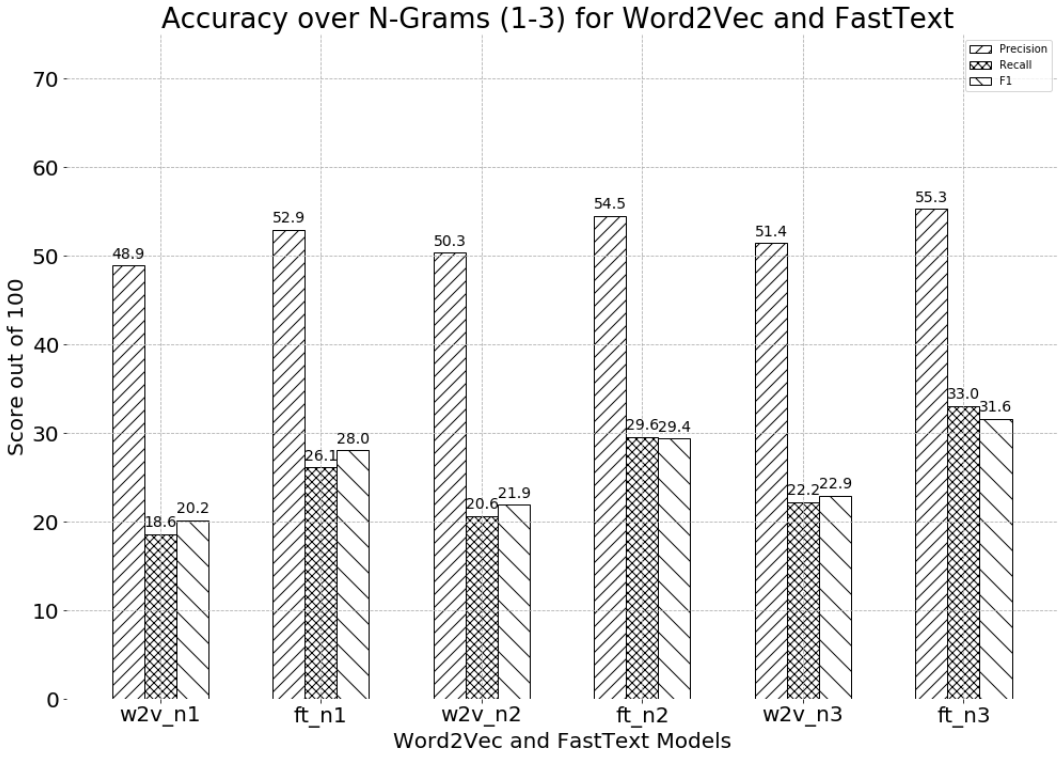

Fig. 4.24. Displays accuracy scores for *Word2Vec (w2v)* and *FastText (ft)* models for n-gram sizes from $n = 1$ to $n = 3$. At every level *FastText* outperforms *Word2Vec* and using more n-grams increases both precision and recall.

dropped every scoring performance metric significantly. For instance, precision, recall and R scores all fell off precipitiously, to around 52.5%, 19.0% and 0.10 for *FastText* using the $n = 3$ n-grams. These same patterns were observed for *Word2Vec* as well. The question is then, why was this pattern observed? One would expect that longer student responses which do not contain words related to the model answer(s) could provide more opportunity for entering false or contradictory information. However, after examining the annotated dataset, it appears that the vast majority of student responses which contain the extra information also receive high scores. This is typically because the extra information is neither contradictory to statements already made by the student which relate to the model answer nor false in anyway, but simply represents additional details that although irrelevant for receiving a high score, are nonetheless correct or at least simply worded or elaborated in more detail than the model answer. For instance, there are 100 items in the dataset which have the label $extra\_info$ and of these 79 have a score $s > 3$, this shows that almost 4 out of every 5 student responses which contain additional details or extra information in some way receive high scores. Therefore, at least on this dataset, it makes little sense to penalize student responses simply for length alone.

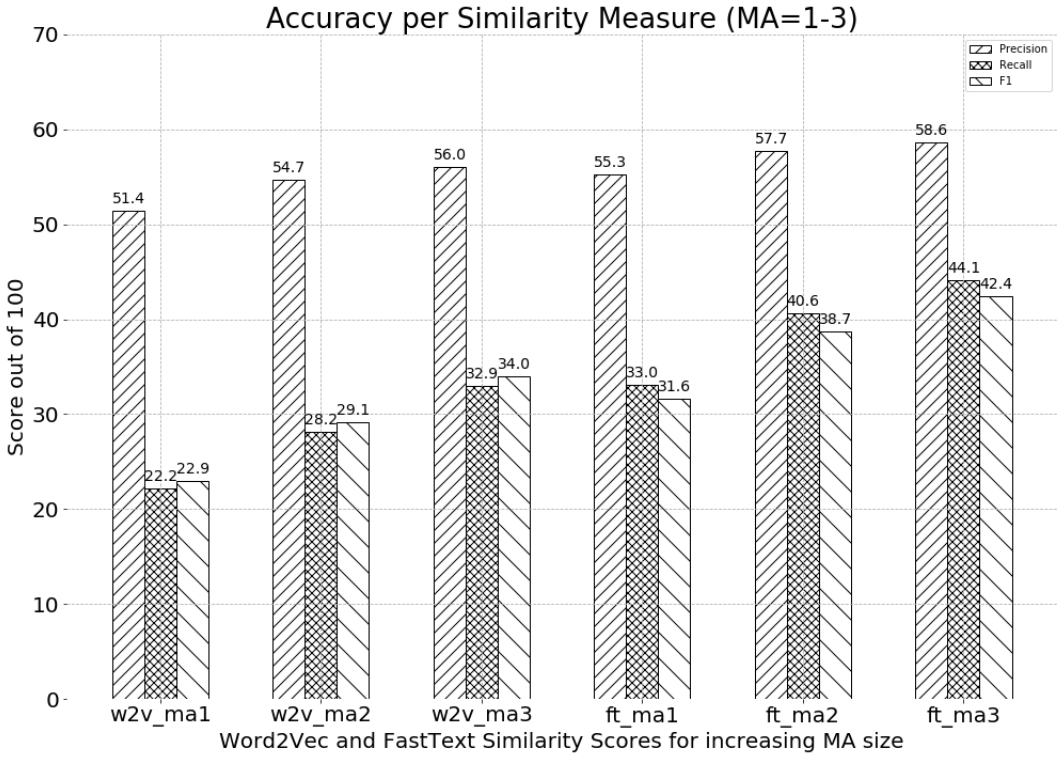

Fig. 4.25. Accuracy Scores for using multiple model answers (MA) *Word2Vec (w2v)* embeddings and *FastText (ft)*embeddings. This demonstrates that increasing model answers improves ASAG performance significantly, albeit with diminishing non-linear returns.

TES.II: Time and Space Complexity

Computationally, word embedding techniques are faster than the graph searches used by knowledge-based approaches because they can use efficient matrix operations to determine the similarity between word vectors as opposed to having to search throughout a graph structure to find node distances. However, word embedding models do take up a significantly larger portion of memory and although these word embeddings can be automatically trained and learned, this process requires a massive amount of text data in order to learn accurate word semantics. Neither of these limitations are concerning for Stembord, however, they are nonetheless attributes which must be taken into account when developing an ASAG solution using word embedding approaches.

TES.III: Amount of Teacher Expertise and Effort Required

Word embedding approaches to ASAG require as little or as much effort as all approaches which only demand that a teacher write in one or more model answers.

TES.IV: Fulfillment of Desired Characteristics for Stembord

Formative Feedback

Word embeddings have the same potential feedback mechanism as all approaches which map from words in a model answer to words in a student response and score based on the semantic similarity of those mappings, namely, that an interface could visually display these mappings and their associated scores. However, although this would increase transparency and understanding of the underlying scoring architecture for users of the system, it does not, in the author's opinion, provide adequate informative feedback for students wishing to understand why they received a particular score since it requires far too much knowledge about technical details.

Works for Self-Study and Teacher-Guided Classrooms

The best performing approach using only word embeddings for the task of pass/fail grading achieves a 74% precision and 77% recall, the details of various measures using different models are shown in the appendix in Tab. A.8. This represents an improvement over the knowledge-based approaches and means that almost 3 out of 4 student responses which are passed by the model should indeed pass. Further, almost 4 out 5 student responses which are failed are bad responses which indeed should have receiving a failing score. This approaches a level which is almost useable in Stembord without being too frustrating to users of the platform.

Effectiveness Across Languages

The word embedding experiments were run for the Spanish language and showed significant improvements over the knowledge-based approaches. However, they did perform worse than the English language word embedding models and there could be numerous reasons for this. The first, comes from the overall lower performance and robustness of NLP tooling in languages other than English for all of the preprocessing tasks. The second, is that the author is not a native speaker of Spanish yet the original dataset was translated from English into Spanish using Google Translate followed by the author correcting some mistakes on his own. This allows errors to enter both from Google and from the human translator. In order to control for all of these variables in the future, the actual percentage performance differences in each NLP tool per language such as POS-Taggers, dependency parsers, etc. would need to be measured and a dataset of questions and model answers would need to be created by a native speaker in the given non-English language. Then the students would need to also be native speakers and write their responses in the non-English language. This is important information to know going forward, since when Stembord decides to implement multiple languages, these factors will have to be controlled for.

Conclusion

The key insight obtained in this experiment is that word embeddings outperform knowledge-based approaches at extracting semantic meaning from text and that, of the word embedding approaches, *FastText* performs significantly better than *Word2Vec* for the task of ASAG. Additionally, it has been reiteratively confirmed that using n-grams of up to size $n = 3$ and increasing model answers are both ways to improve the performance of unsupervised automatic scoring by providing broader coverage of the semantic space. Finally, a key observation is that length penalization is detrimental to automatic scoring performance since it penalizes too much for extra information provided in student responses and this information, at least on this dataset, has a high probability of not containing any errors or reasons for which a human grader would assign a lower score.

4.5 Machine Learning

The purpose of the experiment in this section is to combine all the insights gained in the research up to this point and explore how much ASAG performance can be improved by using a combined and supervised approach. Additionally, insights gained from labeling student responses in the dataset are used in order to determine how much performance could potentially be gained by detecting such attributes of a given student response. For instance, does knowing that a student response misses an important concept - according to the added hand-annotations of the author - increase the ability of a machine learning model to assign a correct score to the student response?

4.5.1 Exp. 1: Classification Scoring with Machine Learning Models

Exp. 1: Background, Reasoning and Hypotheses

It has already been established that using multiple model answers, n-grams of size n up to 3 and *FastText* word embeddings are all the best ways to model automatic short answer scoring in an unsupervised manner. These methods thus provide information for feature engineering in a hybrid machine learning solution.

Initially, many different features and learning algorithms were tried but regardless of the approach, six class scoring could not be learned by a machine learning model, it proved far too difficult for the models to differentiate amoung the various scores between correct (5) and wrong (0). Therefore, 3-class classification was attempted for the classes *wrong*, *partially_correct* and *correct*. This time the data could be learned, however, the models were still not robust enough against valid yet unrelated student responses, since wrong student responses and responses with little to no thematic relation to the question are essentially absent from the dataset. Therefore, the dataset was extended using some unrelated student responses. These were sentences drawn from the Brown corpus. They varied in length and content, but served to simulate a student typing in a sentence or response that is not correct and not thematically related to the topic at hand. Additionally, some wrong answers which were related to the topic were also entered in order to even out the distribution of the dataset. After these measures were taken the trained model seemed to behave reasonably well using sample text inputs from the user.

Importantly however, the evaluation statistics and upper bounds change due to this altered dataset, they are given for this 3-class classification problem in Tab 4.2.

The following describes the features which are used as input to train the model. These are almost entirely taken from the results of the previous experiments.

Features

1. **N-Gram Entailment**: FastText Word Embeddings using entailment scoring and n-grams where $n \in \{1, 2, 3\}$ as detailed in subsection 4.4.2 and shown in Fig. 4.24.

Measure	3-Class Grading
Accuracy	80.10%
Precision	85.12%
Recall	80.10%
F1	81.50%
R	0.79
Kappa	0.65

Table 4.2. New upper bounds for 3-Class Grading (Wrong / Partially Correct / Correct) after extending the dataset.

2. **Cosine Coefficient**: The cosine coefficient word overlap was chosen for its stricter interpretation and scoring of surface structure compared to other techniques such as the faucett coefficient. This is taken directly from the results presented in subsection 4.2.1.
3. **Cosine Coefficient Not**: The same as the above except all negated words are prefixed with a *not_* until a punctuation break is reached. This is a technique used to extract differences between sentences with negation as in *John is a Sailor* and *John is not a Sailor*.
4. **Length Difference**: Difference in length between the model and student answers was chosen to inform the relationship between total information in the *model-answer, student-response* pair and to roughly determine whether or not the student response contains enough detail. This feature was suggested in research done by Sultan et al [SSS16].
5. **BoW Count Vector N-Grams**: Bag-of-Words count vectors made of lemmatized n-grams where $n \in \{1, 2, 3\}$ were used to extract repetitive and ordered surface structure missed by the cosine coefficient.
6. **Token Edit Distance**: measure utilized to extract word order information, this is the suggested best measurement from the research results of section 4.3.
7. **Word Mover Similarity**: used a FastText model to calculate Word Mover Distance, a measure of the amount of distance a student response has to travel to convert its meaning into the meaning of the model answer. Used to measure a numerical similarity score between two sentences or phrases. The *FastText* model is used because of the conclusions drawn from subsection 4.4.2 and the distance metric is one of several suggested by research done by Adams et al [ARK16].

These features are chosen based on their correlation with the human assigned score values, the results many of them have offered in previous experiments and what the literature on the topic suggests. Those features which correlate highly with the gold standard scores are chosen. For example, the token edit distance is chosen because it negatively correlated to the highest degree of all the other syntax measurements explored, higher than the tree, POS and dependency parse edit distances.

Additionally, for comparison purposes two further features are added: *Missed Concept* and *Partially Missed Concept*. These features are not practical or capable of being implemented by a machine at this time but are used in order to determine

how well a hypothetical system could potentially function given the ability to extract these features. They are drawn from the annotated labels assigned in the process of explorative data analysis discusses in section 3.1.3 and are described in Tab. 3.3. The goal is to evaluate how much improvement can be gained by knowing if a student missed a complete concept or just a portion of a concept in a given response.

The models which are evaluated are the Multilayer Perceptron (MLP), Logistic Regression (LR) and Random Forest (RF).

Exp. 1: Presentation of Results

Exp. 1: TES.I-IV

TES.I: Automatic Grade Assignment

As presented in Fig. 4.26, one can see that this approach does offer a performance benefit over previous methods. In the plot on the right one can further see that by adding the two features *Missed Concept* and *Partially Missed Concept* the model performs roughly as well as a human grader. Its unfortunate that at the current time there is no sophisticated way - beyond the pattern matching approach - to determine whether a student response has missed a part or the entirety of a concept.

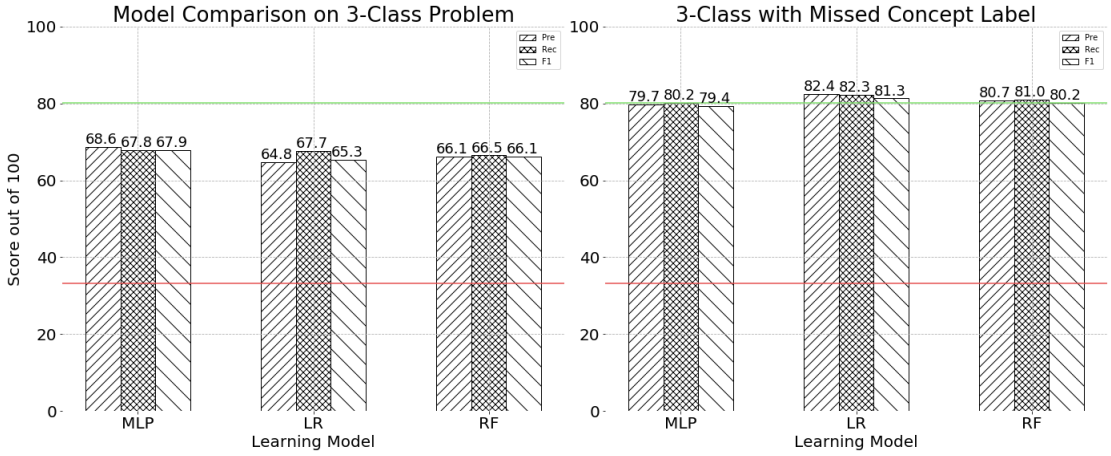

Fig. 4.26. This compares the results using the regular features across all three models vs. the results obtained by adding the *Missed Concept* and *Partially Missed Concept* features. Both show significant improvements over the previously studied approaches, however, by adding the concept features the models are able to attain human level grading capabilities.

TES.II: Time and Space Complexity

This approach is time and space intensive during the training phase. However, during evaluation it is sufficiently fast since creating the features for one *(model-answer, student-response)* pair can be done in the worst performing time of the constituent features and is polynomial. There is an initial startup lag required to load the word embedding models, however, the model needs only to be loaded once and thus, this lag only affects an application's startup time.

TES.III: Amount of Teacher Expertise and Effort Required

This approach also benefits from using multiple model answers. It was found that a modest boost of around 2% could be had by prompting the content creator to entire multiple model answers. However, this boost is not so significant that it warrents a requirement that teachers entire multiple model answers. Therefore, this approach is largely low effort from the teacher's perspective. It is true, however, that they must first hand-grade a set of student responses and only then can the model be trained.

TES.IV: Fulfillment of Desired Characteristics for Stembord

Formative Feedback

This approach doesn't give very informative feedback. It is limited in scope like all other approaches except the pattern matching method, in that it can only show the users of the system whether the student response is correct, wrong, or somewhere in-between.

Works for Self-Study and Teacher-Guided Classrooms

This approach works excellently for *self-study* classrooms in Stembord. A model can be associated with a MOOC and after that point all short-text student responses can be automatically graded. For classrooms with a teacher, the process is initially more difficult since it can only be offered after a first version of the course has been completed and the teacher has already graded a set of student responses.

Effectiveness Across Languages

All of the features used to build this model are easily transferable to other languages, this is mainly because the approach uses word embeddings to extract meaning and does not rely on knowledge based semantic approaches.

Conclusion

The hybrid approach shows marked improvement over the unsupervised methods explored in previous experiments although it only works for 3-class automatic scoring (and consequently pass/fail scoring as well). This experiment shows that *missed concepts* play a vital role in the scoring process. Further, it shows that given

an ability to recognize *Missed Concept* and *Partially Missed Concept* a machine learning model can, in practice, perform roughly equivalently to a human grader for 3-class scoring. Excluding the ability to distinguish missed concepts, there is still a significant amount of room for improvement in the supervised approach. Overall, the choice of machine learning model does not appear to make much of a significant difference. The Multi-layer Perceptron model works best for the standard features, and the logistic regression model is best when the features are extended to use *Missed Concept*. However, these differences are small percentage points and, in the author's opinion, largely negligible.

4.6 Conclusions and Research Informed Development Decisions

The previous research shows that there are roughly three solutions to the problem of building and integrating an ASAG module for Stembord. The first solution uses pattern matching. It allows a teacher to create patterns, as in the regular expression approach examined in the initial experiment. This approach does not appear to work well unless the teacher spends many hours hand-crafting patterns in tandem with viewing an already large list of student responses. However, if this large amount work is done, then this approach has nice user feedback and grades decently quite well, but still misses many correct answers due to the simple variety of human language and inability of human pattern creators to foresee all possible permutations of student responses. The second set of approaches are unsupervised in nature. In them, the teacher simply enters one or more model answers, ideally at last three according to the research. After this initial effort, all subsequent student responses can be automatically graded. This set of methods, however, suffers from several problems, primary amoung them are poor performance and limited feedback. The third approach is supervised. In it the content creator must initially grade a certain amount of student responses, for example, all responses in the first semester of a course offering. Subsequently, a machine learning model can be trained using features extracted from the *(model-answer, student-response)* pairs and the gold standard scores added by the teacher.

This final approach, although initially probably more time consuming for educational content creators, is in the author's opinion much more robust. The regular expression approach, for instance, requires entire days of time and that teachers observe and use student responses in order to create robust patterns which have high enough accuracies to be detect a large swathe of student responses. The unsupervised approaches work at some level, but have recall scores which are too low to not be frustrating for users of the system. Conversely, the supervised approach has the added benefit that it builds a large, well structured, and human graded dataset across different domains, languages, course types and short answer questions and responses. This is essential for more deeply understanding and developing solutions for ASAG in the future. The unsupervised approaches, unfortunately, do not allow for building up a robust dataset or for understanding the process of human grading as applied to short-answer responses and, therefore, are not conducive to long-term iterative improvement of ASAG, at least in these methods' current incarnations. Secondly, using supervised machine learning enables a modelling of grading closer to the way a human approaches the process and allows for future research directions that explore how, for instance, *Partially Correct* student responses can be better differentiated.

For these reasons, it has been decided to use the machine learning model for Stembord as follows. A teacher creates a classroom and a set of quizzes which contain *short-answer* responses. Then students enlist in the course and the teacher must grade all the student *short-answer* responses in the first semester or online version of the class offering. Afterwards, Stembord will export the data, a site ad-

ministrator will train a machine learning model and when completed, associate it with the classroom. Then, for the second version of the class offering, in the next semester at a university for instance, all the student responses can be automatically scored by the system. For *self-study* classes, the approach works even better but is slightly more subtle. In a *self-study* or massive online MOOC classroom, the MOOC creator writes model answers as usual and releases the course. Initially, students begin taking the course and receive automatic short answer grades using the unsupervised word embeddings approach, since the experiments reveal it to be the most effective for unsupervised grading. After an initial critical number of student responses per question has been obtained, the responses are graded by the course creator and then the data is exported, the model trained and, upon completion, associated with the *self-study* classroom, after which all student responses are graded by the machine learning model. Eventually the entire process of exporting data, training a model and associating it again with a course can be automated.

Probably the biggest remaining question after conducting this research is how generalizable the machine learning approach is across different subjects and languages. For instance, would a model trained on computer science questions in English transfer to grade efficiently on biology questions in French? The author's guess would be no since different languages and subjects likely have different structural patterns, but until Stembord gathers enough data across disparate subjects there is no way to answer this question.

5

Software Module Implementation

"Truth can only be found in one place: the code."
— Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship

This chapter describes how the proposed machine learning model which was researched and developed in the previous chapter, is integrated into the Stembord LMS. The architecture of this implementation can be broken down into the following two categories:

1. **Stembord Components**: the design of internal software classes and architecture of the Stembord application itself, specifically those parts relevant to the integration of the ASAG module.
2. **Quiz API**: the modular external service designed for advanced or complicated question types such as short-answer. This is the main piece of software managing the ASAG service in Stembord.

These two components coordinate to provide Automatic Short Answer Grading in Stembord and are described in detail in this chapter.

5.1 Stembord Components

Stembord Architecture and System Design

Stembord is programmed in the Elixir programming language. Elixir is a programming language built on the Erlang virtual machine, which is designed for highly distributed and fault-tolerant systems [Pla18]. A MVC based web framework programmed in Elixir known as Phoenix is used to implement the backend server API. The frontend GUI of the website is programmed in JavaScript, HTML and CSS.

The main components of Stembord as well as those which are important for understanding the implementation of the ASAG module are outlined in this section. Those of most importance for understanding the implementation of the ASAG module are the system for managing Learning Objects (LOs) as well as the node grading subsystem.

The main components of Stembord are shown in Fig. 5.1.

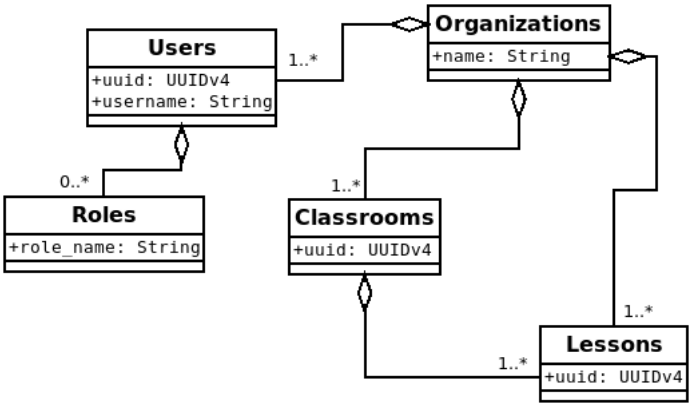

Fig. 5.1. The Users, Roles and Organization structural components of Stembord.

Every user of the system can be either a *Student*, a *Teacher* or both, just like in the real-world. This state is modeled by assigning a user a role having a *role_name* attribute of *teacher*. All users on the platform have access privileges and the abilities of students. Student is the default role. An *Organization*, for example a university, has many users, some of them are students, some are teachers and some are both. Users with a *teacher* role can create *Classrooms* and *Lessons* for an *Organization*. A *Classroom* has many *Lessons* but *Lessons* can also be unassociated with a *Classroom* and simply belong to the *Organization*, a typical example of this scenario would be a single lecture or tutorial that a teacher would like to give on a particular topic independent of any specific class. *Lessons* and *Classrooms* can be made publicly accessible and then any student on the Stembord platform, even those which do not belong to the organization, can study the material for themselves. Further, any teacher can copy the *Lesson* or *Classroom*, change it to fit his or her needs and then reuse it to teach a class or give a lecture.

Lessons in Stembord

The lessons module is the locus of core functionality on the Stembord platform, more precisely, it is the module which handles LOs such as quizzes and a myriad of learning content objects, for instance, texts, questions and interactive media. Figure 5.2 demonstrates the class structure of a section of this module.

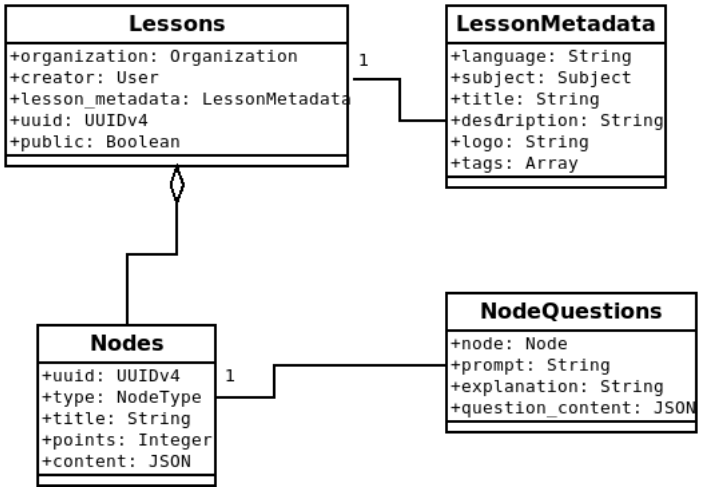

Fig. 5.2. The structure of LOs on Stembord. Lessons are made up of nodes. Lessons have metadata associated with them and a node can have a node question data structure if its *NodeType* attribute designates it as being a question node.

Lessons belong to an *Organization* and can be associated with a *Classroom* through a join table making them independent of, but usable by, the classroom system. *Lessons* have metadata which stores the language of the lesson, for instance, English and a subject which indicates the topic of the lesson, for instance, computer science or biology. Lessons also have a UUID for globally unique access and a boolean field *public* which when true allows other teachers on the platform to copy the lesson and other students to see, read and consume it.

All lessons are made up of a collection of nodes which are linked to one-another in an ordered list fashion. Every node has content, which is a JSON formatted string holding the information in a structured machine processable format for rendering in the frontend web application. Nodes also store points which can range from 0 to 1000 and can be used by teachers to give grades for answering questions and completing quizzes. Each node has a *NodeType* which designates what data is stored on the node and how the node behaves. There are currently node types for text and questions. A text node type can not only render any formatted text but also most mathematical latex, embedded images and URL links. This makes it flexible as a content type for creating LOs. When a node of type question is created an associated *NodeQuestion* object is instantiated and a corresponding entry is stored in the database and associated with the node. This *NodeQuestion* object can store a question prompt, an explanation of the correct answer, and a

JSON formatted content text which varies by question type. The available question types are *multiple choice*, *true/false*, and after now *short-answer*.

The classes in Fig. 5.2 are sufficient to store, display and consume LOs such as guided tutorials, quizzes and lecture videos, however, in order to provide educators with summative and formative assessment capabilities other structures are necessary. These are shown in Fig. 5.3.

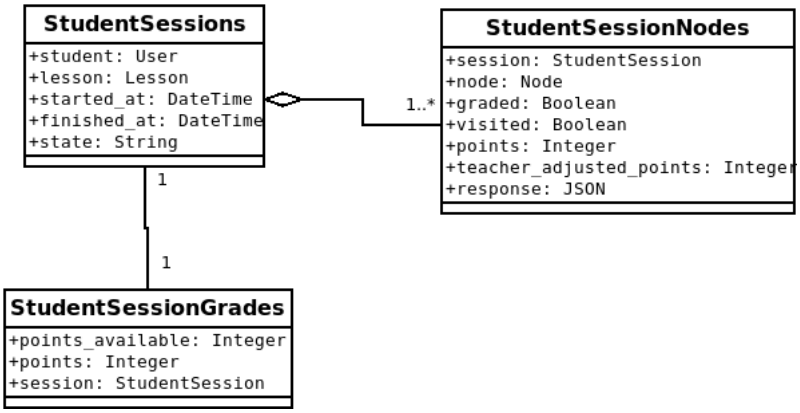

Fig. 5.3. The Grading System in Stembord - StudentSessions, StudentSessionNodes and StudentSessionGrades

When a student starts working on a *Lesson*, Stembord creates a *StudentSession* object whose state is set to *in-progress*. As the student then works through the lesson and completes nodes such as watching a video node, reading a text node or completing some question nodes, *StudentSessionNodes* are created which store the responses of the student for that particular node in the session. Once the student has finished the lesson the *StudentSession* object's state is set to *finished* and the object is visible to teachers in an interactive grade editor. Teachers can then review and adjust the student's points on a per node bases, if they so desire, and when they are finished, they mark the student's session as graded by clicking a labeled button in the editor which creates a StudentSessionGrade object and stores the student's final grade for the LO in question.

The Node Grading Subsystem

Fig. 5.4 shows the student session view of a Multiple-Choice question node as just discussed in the previous section. When the user presses the check button his answer is submitted to the backend web application where it is graded and finally returned to the user to show him the results of his choice, such as whether it is correct and how many points he receives. This section discusses that entire process.

There is one endpoint which handles scoring nodes in Stembord, it is accessible via an HTTP POST request to the url `/session/nodes/:id/answer`, where *:id* is the identifier of the node object. The frontend of Stembord as seen in Fig. 5.4 makes a request to the backend application server at this URL, sending the ID

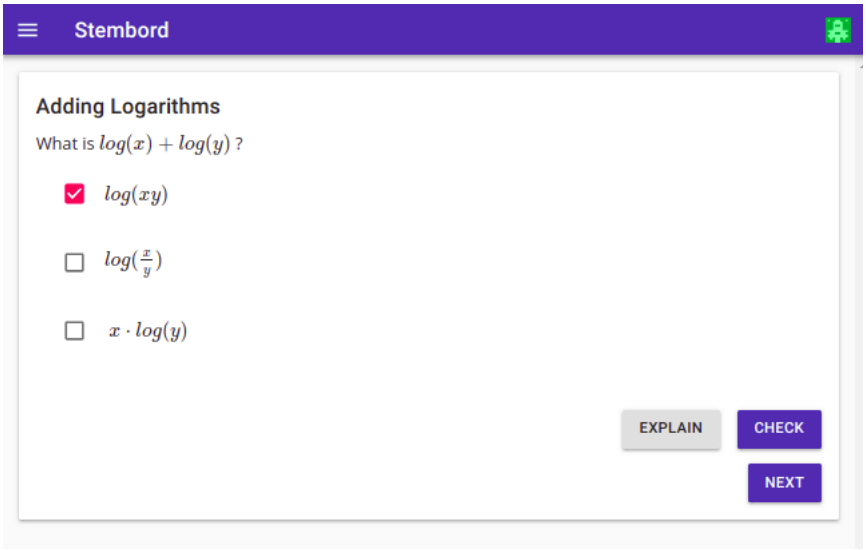

Fig. 5.4. The Stembord application view. This is the student's view inside a student session. He is getting ready to submit his answer to a Multiple-Choice question node by clicking the *Check* button.

of the node along with the student's answer in the request body. The Phoenix Framework handles parsing the URL and the parameters and dispatching this request object to a controller method *answer()* which is shown in full in Lst. 5.1. This method fetches the session object and actual node from the database and passes the session, node and student's answer to the *NodeGrader.grade_item* method call. The score obtained from this call is returned to the frontend as a JSON response where it is then rendered for the student.

Listing 5.1. The answer method inside the student session controller. This gets the active session for the student as well as the current node from the database. It then passes this information on to the NodeGrader for actually grading the student's response.

```
1  defmodule StembordWeb.Students.StudentSessionController do
2
3      def answer(conn, %{"id" => id, "answer" => answer}) do
4
5          # fetch the student user and current lesson
6          user = conn.assigns[:user]
7          lesson = conn.assigns[:lesson]
8
9          # fetch the session and the node from the database
10         with {:ok, %StudentSession{} = student_session} <-
11                  StudentSessionService.get_session(user, lesson),
12              %Node{} = node <- NodeService.get_by(lesson.id, id),
13              {:ok, %StudentSession{} = student_session} <-
14
15                  # grade the student's answer inside the current session
16                  NodeGrader.grade_item(student_session, node, answer) do
17
18         # render the result as JSON
19         conn
20         |> render("show.json", student_session: student_session)
21
```

```
22          end
23      end
24  end
```

The *NodeGrader.grade_item* function always receives three objects, the student's current session, the question node within that session and the answer the student has submitted. Within a database transaction, it then grades the answer, stores the result on the session node, marks the node as completed and returns the updated session object to its caller. This process is shown in Lst. 5.2

Listing 5.2. The function which grades each question node and updates the database entries.

```
1  defmodule Stembord.Services.NodeGrader do
2
3      def grade_item(
4              %StudentSession{id: student_session_id},
5              %Node{id: node_id, question: question, points: points},
6              user_input
7              ) do
8
9          # begin a database transaction
10         Repo.transaction(fn ->
11
12             # get the student session node
13             student_session_node =
14             StudentSessionNode.Api.get_by!(
15                 student_session_id: student_session_id,
16                 node_id: node_id)
17
18             # compute the raw grade
19             raw_grade = score_question(question, user_input)
20
21             # store the scaled and raw grades on the
22             # student session node
23             StudentSessionNode.Api.update!(student_session_node, %{
24                 student_completed: true,
25                 student_grade: raw_grade,
26                 student_points: Kernel.round(raw_grade * points)
27             })
28
29             # fetch the updated student session from the database
30             # and return it
31             StudentSessionService.get!(student_session_id)
32         end)
33     end
34 end
```

Finally, the method *score_question()* needs to be discussed. It uses Elixir's pattern matching capabilities to dispatch to a different function based on the question type of the given node. The code displayed in Lst. 5.3 shows how this is implemented for the simple True-False question type.

Listing 5.3. field. If this is true a perfect 1.0 is returned. Otherwise a failing 0.0 is returned.]TThe score question function matches on NodeQuestion objects which have a question_type attribute of :true_false. It then passes the parameters down into the *is_true_false_correct* function, which checks to see if the student's answer is the same as the answer the teacher has set in the question_content["answer"] field. If this is true a perfect 1.0 is returned. Otherwise a failing 0.0 is returned.

```elixir
defmodule Stembord.Services.NodeGrader do

    def score_question(
            %NodeQuestion{question_type: :true_false,
                          question_content: question_content},
            user_input
        ) do
        is_true_false_correct(question_content, user_input)
    end

    # ...

    def is_true_false_correct(question_content, user_input) do
        if question_content["answer"] == user_input do
            1.0
        else
            0.0
        end
    end

end
```

The flow of information is the exact same for other question types except that other types have their own corresponding score_question and implementation functions. There are particular versions for *Multiple-Choice* as well as *Short-Answer* question node types, however, they are much longer and do not provide extra information about the architectural design.

For short-answer, the question type developed in this thesis, its *score_question* function makes a call to an external API service written in Python. The reasoning behind this is two-fold. First, Elixir cannot do text processing well and Python is a much better language with many more robust libraries for handling all the text processing needs of such a module. The second reason is that there is a lot of code required to implement short-answer scoring and possibly other complicated question types in future versions, so for the sake of modularity and scalability it was decided to externalize complicated question types to their own API service. This external API service is described in the following section.

5.2 Quiz API

An external REST API was programmed in Python using the Django Framework [Fou18] in order to offer ASAG grading services to the Stembord backend application in a decoupled and modular manner. The following sections discuss the layout, design and core classes implemented by this API.

5.2.1 Rest API Architecture

At the time of this writing, the REST API offers only two URL Endpoints which are listed below.

1. **Short-Answer**: `/api/question_types/short\_answer`
2. **Fill-In-The-Blank**: `/api/question_types/fill_in_the_blank`

These two endpoints taken together allow for automatic grading of short answer responses. *Fill-In-The-Blank* is an alternative created to allow teachers to model the very short short-answers discovered during explorative data analysis in subsection 3.1.3. *Short-Answer* handles all other automatic short-answer response grading.

The quiz API is designed with a modular architecture allowing for easy decoupled development and maintenance. View handlers delegate incoming requests to underlying modules designed to grade specific question types in quizzes, for instance, *short-answer*. Fig. 5.5 shows a conceptualization of the information flow for the *short-answer* endpoint.

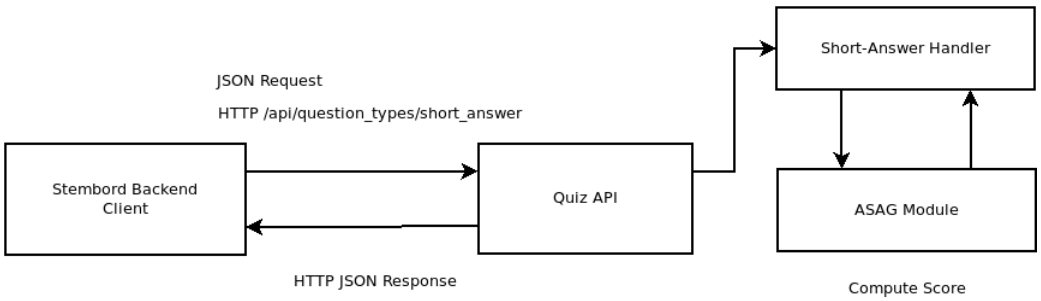

Fig. 5.5. Demonstrates architectural flow of information. The Stembord Backend Client makes an HTTP request to the Quiz API Server in order to grade a *short-answer*. The server dispatches the request to the *ShortAnswerHandler* class which checks and validates the input parameters before sending the information further down the pipeline to the ASAG Module itself. The ASAG Module computes the score given the input and sends the results back to the handler. The handler then sends this information back to the Quiz API and Django handles sending the response JSON object to the calling service, in this example the Stembord Backend Client.

The flow of information from request to handlers, to question module and back up the pipeline remains the same for every other endpoint in the API. Each endpoint does, however, require a different set of parameters, specific to the endpoint,

at its interface to the calling client. The returned response is always a JSON object consisting of a standardized set of fields. The contents of this returned JSON response object are shown in Lst. 5.4. The response contains grading scores, feedback and debugging information.

Listing 5.4. Quiz API JSON Response. Every endpoint returns three values: a score, split up into raw and scaled scores, a potentially non-empty array containing feedback information such as that the student response contains a spelling mistake, and a log field which holds debugging information.

```
response = {
    'score': {
        'raw': 0.8,
        'scaled': 16
    },
    'feedback': ['spelling error'],
    'log': []
}
```

Since this thesis is about researching and developing an automatic short answer grader, the details of the *Fill-In-The-Blank* module are not discussed further, but instead a focus is made on the *Short-Answer* module. The *ShortAnswerHandler* class requires that its clients send several parameters which are important for further processing in the pipeline. These parameters can be seen in Lst. 5.5.

Listing 5.5. Parameter schema for *ShortAnswerHandler* class. A model answer, student answer, points_available, and method are required parameters. The *points_available* parameter holds the points which a student can achieve if he answers 100% correctly. Its an integer value, for instance, 10. The *method* allows the caller to choose an ASAG approach, for instance, *Cosine_Coefficient* could be chosen instead of machine learning. Some of these approaches have various options such as whether to use stemming or lemmatization as a preprocessing step. These are given by the *method_options* parameter. Finally, the language can be set although it defaults to English and the API currently only offers English.

```
    s = Schema()
    s.field('model_answer', Type.STRING, required=True)
    s.field('student_answer', Type.STRING, required=True)
    s.field('points_available', Type.INTEGER, required=True, coerce=True)
    s.field('method', Type.STRING, required=True)
    s.field('method_options', Type.STRING)
    s.field('language', Type.STRING)
    s.field('debug', Type.BOOLEAN)
```

The full handler implementation for short-answers is shown in Lst. 5.6.

Listing 5.6. Implementation of the *ShortAnswerHandler* class. First, the language is set, then an ASAG scoring class is created for the given method, for instance, *Cosine_Coefficient* or *Machine_Learning*. This scorer then scores the *(model-answer, student-response)* pair returning a raw score. The raw score is then scaled and normalized and a response is constructed as seen in Lst. 5.4. If the debug parameter has been set, a log of the details of the preprocessing and scoring processes which occur inside the *scorer.score()* call are returned.

```
1  class ShortAnswerHandler(AbstractHandler):
2
3      def call(self):
4
5          # set the correct language
6          i18n.set('locale', self.settings['language'])
7
8          # create the scorer given the method
9          scorer = create_scorer(self.settings)
10
11         # score the questions
12         raw_score = scorer.score()
13
14         score_normalizer = ScaleAndRoundScorer(
15             self.settings['points_available'])
16         scaled_score = score_normalizer.scale(raw_score)
17
18         # build the response
19         response = create_response(raw_score, scaled_score)
20
21         # append the log if requested
22         if self.in_debug_modus():
23             response['log'] = self.get_log()
24
25         return response
```

In summary, the client sends an HTTP Post request with the required parameters. These parameters are then validated and parsed by the handler, this information is then passed down to the ASAG module or scorer, which in turn, instantiates the correct class for the given method parameter and its options. The scorer class then computes the scores and returns them to the handler. The next section describes the subprocess which takes place inside the ASAG scorer.

5.2.2 ASAG Scorer Module

The scorer module is instantiated by a *create_scorer()* function shown in Lst. 5.7. This function checks which methodological scoring approach the client has requested and instantiates a class to score the *(model-answer, student-response)* pair using that particular approach. If, however, an invalid approach has been chosen an exception is thrown and an error is displayed to the client indicating that the request was invalid.

Listing 5.7. The *create_scorer()* function. It instantiates an ASAG object which handles scoring short-answer responses in a particular way. In this code example, the two options are the "cosine_coefficient" approach examined in Exp. 4.2.1 and the hybrid machine learning model approach from Exp. 4.5.1. These are imple-

mented by the *CosineCoefficientModel* and *MachineLearningModel* classes respectively.

```
def create_scorer(settings):

    asag_method = settings['method']

    if asag_method == 'cosine_coefficient':
        return CosineCoefficientModel(settings)
    elif asag_method == 'machine_learning_model':
        return MachineLearningModel(settings)
    else:
        raise Exception("unrecognized ASAG method: {}".format(asag_method))
```

Each approach has its own set of options it can accept and its own class which handles the internal details of that specific way of doing ASAG. For the purpose of simplicity and illustration the *CosineCoefficientModel* class is examined, however, each of the general steps applies to the other scorer classes as well such as the *MachineLearningModel*.

Every ASAG class inherits basic functionality from the *BasicModel* class and must implement its abstract score function. The CosineCoefficientModel class, which does just this, is shown in detail in Lst. 5.8.

Listing 5.8. The ASAG score function as implemented for the *cosine_coefficient* automatic scoring method. The *(model-answer, student-response)* pair is preprocessed, debug information is logged if the debug parameter has been set, the cosine coefficient score is calculated and finally the raw score is returned with $1e-300$ added in the denominator to prevent division by zero exceptions.

```
class CosineCoefficientModel(BasicModel):

    def score(self):

        ma = self.params['model_answer']
        sa = self.params['student_answer']

        # preprocess the model answer
        ma_t = word_overlap_preprocess(ma)
        ma_ts = set(ma_t)

        # preprocess the student response
        sa_t = word_overlap_preprocess(sa)
        sa_ts = set(sa_t)

        if self.in_debug_modus():
            self.handler.log_info(
                "[model answer preprocessed]: {}".format(ma_t))
            self.handler.log_info(
                "[student answer preprocessed]: {}".format(sa_t))
            self.handler.log_info(
                "[model answer as set]: {}".format(ma_ts))
            self.handler.log_info(
                "[student answer as set]: {}".format(sa_ts))

        # calculate the cosine coefficient score
        num = len(ma_ts.intersection(sa_ts))
        den = np.sqrt(len(ma_ts)) * np.sqrt(len(sa_ts))

        # prevent ZeroDivisionError
        return num / (den + 1e-300)
```

As seen in the handler implementation in Lst. 5.6 this raw score is then scaled and returned in the JSON response object back to the client caller.

6

Outlook

> "Wisely and slow. They stumble that run fast."
>
> — William Shakespeare in Romeo and Juliet, Act 2, Scene 3

From the standpoint of Stembord, the next steps for automatic short answer grading are as follows. First, much more short answer data must be gathered across as many topics and different classrooms and languages as possible in order to determine to what degree the insights made in this thesis hold up on these new and different datasets, each having different scoring distributions, human graders providing supervised gold standard scores, and covering vastly different subjects. These subjects will still include computer science but extend to chemistry, biology and even literature. Further, the necessity of collecting and building the data store is essential and, therefore, Stembord has decided to implement a specific set of guidelines, which require teachers to manually grade all short answer questions in the first version of an online classroom. This has the objective of building the large and diverse dataset needed to further evaluate and expand upon the results obtained in this thesis.

In terms of the first version of the ASAG solution implemented in this thesis, one can rather accurately assert that it is quite limited in scope, since it is still well under the hypothesized ideal solution. This makes it primarily only useful for *self-study* classrooms or low-stakes quizzes. Further research should examine new features and techniques, especially those which have shown promise in recent years such as deep learning and, specifically, LSTMs. These methods can be used in order to increase the robustness of models in terms of precision and recall and to hopefully model the signals of the underlying student responses in a more accurate manner. However, based on the results in this thesis and of special importance, are techniques which can allow for finer grained insight into the classification of *Partially_Correct* student responses and especially those methods which help differentiate the edge zones between *Wrong* and *Partially_Correct* and *Partially_Correct* and *Correct* responses.

Additionally, there are several key questions which have arisen out of the research conducted in this thesis. They are:

1. Can a machine learning model trained on data for a particular subject, for instance computer science, simply be used and applied in a course on a completely different topic, for instance biology?
2. Can a machine learning model trained on data for a particular language, for instance English, be used in a course on the same subject but in a different language, for instance Spanish?
3. What are ways to begin detecting missed concepts and partially missed concepts in student responses?

Answering the first two of these questions will have significant repercussions on the implementation details and approach of automatic scoring within Stembord. Clearly, it would be wonderful to have a single trained model which could automatically score all short answer responses regardless of language and subject matter, however, this does seem unlikely since there are at bare minimum specific word embeddings that will likely need to be learned for correctly constructing the semantic space of the terminology used in subjects such as mathematics, biology and physics. The last question is much broader and more general in scope, but is based on the observation that by using the hand labelled information about partially and completely missed concepts, the trained model is capable of classifying at human grader levels and of distinctly differentiating between *Wrong*, *Partially_Correct* and *Correct* target classes. Being able to get to that level of performance would significantly increase the user experience and allow teachers to use the short-answer grader for some higher stakes tests and quizzes, not just low-stakes learning objects.

Of equal importance to the areas listed above, is examining approaches to providing more detailed and insightful formative feedback both for students and educators, perhaps by developing a grading rubric and scoring short-answer responses according to a rubric allowing for more informative feedback to users.

Other immediate research steps are to look into the effectiveness of the developed ASAG module in a high school classroom in the United States in order to assess how students and teachers respond to the ability to use *short-answer* low-stakes assessments in the classroom and determine whether it can improve student outcomes compared to other question types or compared to students using traditional pen and paper.

The author has made many observations while striving to achieve the goals set out in this thesis and although this has at times been a difficult process, it has also proven enlightening and has spawned many ideas for other research directions. One of these interesting directions would be to look more deeply into the natural language logic and reasoning branch of artificial intelligence. This was partially done while researching for this thesis, however, much fruitless time was spent in the prototyping phase and that direction was eventually disgarded due to complexity and lack of results. More specifically, this approach attempts to extract concepts from text and integrate them with knowledge bases and logic and reasoning systems in order to truly understand student responses, which proved extremely difficult and exploded outside the scope of this thesis. Nonetheless, if a

system could extract concepts from text in a robust and reliable manner then these concepts could be used to grade in a much more transparent and informative way. For instance, a system could offer help to a user on a specific aspect of a topic when that user misses the corresponding portion in his *short-text* response. This would make the feedback from such a system much more human friendly and informative for all users.

In the author's opinion, robustly solving the automatic short answer grading problem requires a system capable of truly understanding the contents of texts and having the ability to reason and manipulate symbols at near human levels, augmented of course by large knowledge bases of domain specific data. This logic, reasoning and deep integration with comprehensive knowledge bases are areas within artificial intelligence which must be explored in order to build systems which can grade in a user-friendly and feedback oriented way, transparently, just as the best and most passionate teachers do.

This thesis has given evidence that grading short-answer questions is a difficult, time consuming process that is more effective than many other question types for internalizing knowledge into students' minds. It has discovered many of the intricate peculiarities and structures which arise in short-answer responses. It has explored and illustrated a list of methods for automatically grading these responses both in a supervised and unsupervised manner. It has shown that under certain situations this process can be quite effectively automated in order to relieve teachers' workloads and achieve better learning outcomes for students. Many of these discoveries have been implemented within a living and breathing software ecosystem so that users can profit from them. And although much work remains to be done, a first step has been made down a long and winding tunnel, where a flickering light far off into the distance has illuminated the next steps along the way. The author can only hope that flicker is daylight and that despite his biases and human errors, his direction is correct.

References

ARK16. Oliver Adams, Shourya Roy, and Raghuram Krishnapuram. Distributed vector representations for unsupervised automatic short answer grading. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications*, pages 20–29. The COLING 2016 Organizing Committee, 2016.

BGJM16. Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information, 2016.

BGS14. Steven Burrows, Iryna Gurevych, and Benno Stein. The eras and trends of automatic short answer grading. *International Journal of Artificial Intelligence in Education*, 25(1):60–117, 2014.

Bil05. Philip Bille. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239, 2005.

BJ99. Lu Chi Burstein Jill, Wolff Susanne. Using lexical semantic techniques to classify free-responses. In Viegas Evelyne, editor, *Breadth and Depth of Semantic Lexicons. Text, Speech and Language Technology*, volume 10, pages 227–244. Springer, Dordrecht, 1999.

Bla18. Inc. Blackboard. Short answer questions — blackboard help, 2018.

Blo84. Benjamin S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13(6):4, 1984.

CF17. Liying Cheng and Janna D. Fox. *Assessment in the language classroom: teachers supporting student learning*. Palgrave in the UK is an imprint of Macmillan Publishers Limited, 2017.

Cjb05. Hamish Coates, Richard james, and Gabrielle baldwin. A critical examination of the effects of learning management systems on university teaching and learning. *Tertiary Education and Management*, 11(1):19–36, March 2005.

Coh60. Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

Dat18. Inc DataCamp. Learn r, python and data science online — datacamp, 2018.

DNB+13. Myroslava Dzikovska, Rodney Nielsen, Chris Brew, Claudia Leacock, Danilo Giampiccolo, Luisa Bentivogli, Peter Clark, Ido Dagan, and

Hoa Trang Dang. Semeval-2013 task 7: The joint student response analysis and 8th recognizing textual entailment challenge. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 263–274. Association for Computational Linguistics, 2013.

FLCP03. Joseph L. Fleiss, Bruce Levin, and Myunghee Cho Paik. *Statistical Methods for Rates and Proportions*. John Wiley and Sons, Inc, 3 edition, 2003.

Fou18. Django Software Foundation. Django, 2018.

Fri08. Jeffrey E. F. Friedl. *Mastering regular expressions*. OReilly, 2008.

GCHO05. A.c. Graesser, P. Chipman, B.c. Haynes, and A. Olney. Autotutor: An intelligent tutoring system with mixed-initiative dialogue. *IEEE Transactions on Education*, 48(4):612–618, 2005.

GHW12. Fahmy A. Aly Gomaa H. Wael. Short answer grading using string similarity and corpus-based similarity. *International Journal of Advanced Computer Science and Applications*, 3(11), 2012.

GPJ18. Palash Goyal, Sumit Pandey, and Karan Jain. *Deep learning for natural language processing: creating neural networks with Python*. Apress, 2018.

Hea00. M.a. Hearst. The debate on automated essay grading. *IEEE Intelligent Systems and their Applications*, 15(5):22–37, 2000.

Hyn18. Brendon Hyndman. Ten reasons teachers can struggle to use technology in the classroom, 9 2018.

Inc18. Kaggle Inc., 2018.

JC97. Jay Jiang and David Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference Research on Computational Linguistics*. Association for Computational Linguistics, 1997.

JM09. Sally Jordan and Tom Mitchell. e-assessment for learning? the potential of short-answer free-text questions with tailored feedback. *British Journal of Educational Technology*, 40(2):371–385, 2009.

JT16. Heit Jamey Jobbitt Todd, Donaldson Robin. The impact of automated assessment on student outcomes: A hangkuk university case study, 2016.

Jur18. Dan Jurafsky. Regular expressions, text normalization, edit distance, 2018.

KA18. Inc Khan Academy. Khan Academy free online courses, lessons and practice, 2018.

Koh99. Alfie Kohn. From degrading to de-grading. *High School Magazine*, March 1999.

KSKW15. Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. From word embeddings to document distances. In *Proceedings of the 32Nd International Conference on International Conference on*

Machine Learning - Volume 37, ICML'15, pages 957–966. JMLR.org, 2015.

Kuk. Karen Kukich. Techniques for automatically correcting words in text.

LC98. Claudia Leacock and Martin Chodorow. Combining local context and wordnet similarity for word sense identification. In *WordNet, An Electronic Lexical Database*, chapter 11, pages 265–284. The MIT Press, 1998.

LS04. Claudia Leacock and Educational Testing Service. Scoring free-responses automatically: A case study of a large-scale assessment. *Examens*, 2004.

MBM11. Michael Mohler, Razvan. Bunescu, and Rada Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, EdAppsNLP 05, pages 752–762. Association for Computational Linguistics, 2011.

MCCD13. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

MM09. Michael Mohler and Rada Mihalcea. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 567–575, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Moo18. Moodle. Short-answer question type - moodledocs, 2018.

MS08. Christopher D. Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT, 2008.

MZOK11. Detmar Meurers, Ramon Ziai, Niels Ott, and Janina Kopp. Evaluating answers to reading comprehension questions in context: Results for german and the role of information structure. In *Proceedings of the TextInfer 2011 Workshop on Textual Entailment*, TIWTE '11, pages 1–9, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

NMB13. Roger Nkambou, Riichiro Mizoguchi, and Jacqueline Bourdeau. *Advances in Intelligent Tutoring Systems*. Springer Berlin, 2013.

oT18. University of Tübingen. License agreements for germanet, 2018.

Pag66. Ellis B Page. The imminence of grading essays by computer. *The Phi Delta Kappan*, 47(5):238–243, January 1966.

PAP+17. Feddy Pribadi, Teguh Adji, Adhistya Permanasari, Anggraini Mulwinda, and Aryo Baskoro. Automatic short answer scoring using words overlapping methods. In *AIP Conference Proceedings*. American Institute of Physics, 2017.

Pea18. Glen Pearsall. *Fast and effective assessment: how to reduce your workload andimprove student learning*. ASCD, 2018.

Ped10. Ted Pedersen. Information content measures of semantic similarity perform better without sense-tagged text. In *Human Language Tech-*

nologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 329–332, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

Pla18. Plataformatec. Elixir, 2018.

PRWZ01. Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL 02*, 2001.

PS05. Stephen G. Pulman and Jana Z. Sukkarieh. Automatic short answer marking. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, EdAppsNLP 05, pages 9–16, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

PVG+11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

RND15. Shourya Roy, Y. Narahari, and Om D. Deshmukh. A perspective on computer assisted assessment techniques for short free-text answers. *Computer Assisted Assessment. Research into E-Assessment Communications in Computer and Information Science*, page 96–109, 2015.

SGAM+15. Grigori Sidorov, Helena Gomez-Adorno, Ilia Markov, David Pinto, and Nahun Loya. Computing text similarity using tree edit distance. *2015 Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC)*, 2015.

SSS16. Md Arafat Sultan, Cristobal Salazar, and Tamara Sumner. Fast and easy short answer grading with high accuracy. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

TMA02. Peter Broomhead Tom Mitchell, Terry Russell and Nicola Aldridge. Towards robust computerised marking of free-text responses. *Proceedings of the 6th CAA Conference*, 2002.

WP94. Zhibiao Wu and Martha Palmer. Verb semantics and lexical selectio. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 1994.

WRW07. Sunnie Lee Watson. William R. Watson. An argument for clarity: what are learning management systems, what are they not, and what should they become?. *TechTrends*, 48(4):28–34, 2007.

YA06. Jill Burstein Yigal Attali. Automated essay scoring with e-rater® v.2. *Journal of Technology, Learning, and Assessment*, 4, 2006.

YHZ+18. Xi Yang, Yuwei Huang, Fuzhen Zhuang, Lishan Zhang, and Shengquan Yu. Automatic chinese short answer grading with deep autoencoder.

Lecture Notes in Computer Science Artificial Intelligence in Education, page 399–404, 2018.

A

Appendix

A.1 Glossary

ASAG	Automatic Short Answer Grading
LMS	Learning Management System
ITS	Intelligent Tutoring System
MOOC	Massive Open Online Course
STS	Short Text Similarity
RTE	Recognizing Textual Entailment
CAA	Computer Assisted Assessment
NLP	Natural Language Processing
SAG	Short Answer Grading
NLU	Natural Language Understanding
ETS	Educational Testing Services
LSA	Latent Semantic Analysis
SVD	Single Value Decomposition
SVM	Support Vector Machine
MLP	Multi-layer Perceptron
NER	Named Entity Resolution or Named Entity Recognition
CoMiC	Comparing Meaning in Context project
GCSE	General Certificate of Secondary Education
BoW	Bag-of-Words
IE	Information Extraction
CAI	Computer-Assisted Instruction
CBI	Computer-Based Instruction
CAL	Computer-Assisted Learning
CMS	Content Management System
LCMS	Learning Content Management System
LO	Learning Object
ECDF	Empirical Cumulative Distribution Function
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
CSV	Comma-Seperated Values (File)
KB	Knowledge Base
STS	Short Text Similarity
MVC	Model-View-Controller
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets

A.2 Tables

A.2.1 Datasets

Labels

Label	Wrong	Partially_Correct	Correct
Missed Concept	61.3%	54.55%	0.07%
Confusing Wording	4.48%	3.67%	0.09%
False Assertion	27.70%	11.58%	0.03%
Lack of Knowledge	5.50%	0.01%	0.0%
Other Wording	0.02%	4.11%	17.52%
Question Misinterpretation	5.10%	2.05%	0.0%
Misspelling	2.44%	2.93%	3.31%
Extra Info	3.26%	4.99%	5.52%
Not Enough Info	6.31%	30.35%	0.55%

Table A.1. Percentages of student responses having a particular label grouped by score category. *Other Wording* is a factor in many correct student responses, there it indicates that the response simply has different words, terms and phrases than the model answer but is nonetheless correct. Besides *Missed Concept*, *Not Enough Info* is the second most common label for partially correct responses, indicating that many responses to not provide enough detail for a human grader to draw an accurate conclusion as to the knowledge state of the student.

A.2.2 Experiments

Regular Expressions

Experiment 1: Teacher Created Regular Expressions

	SE_1	SE_2	HST_1	SE_{1ext}
Accuracy	15.71%	11.27%	15.40%	39.68%
Precision	39.53%	36.37%	38.97%	43.21%
Recall	15.71%	11.27%	15.40%	39.68%
F1	20.21%	14.60%	19.84%	36.78%

Table A.2. Accuracy, Precision, Recall and F1 measurements per respondent.

	P_a	P_{se}	P_t	P_p
Accuracy	33.81%	28.73%	32.38%	63.18%
Precision	86.44%	86.51%	86.95%	88.43%
Recall	33.81%	28.73%	32.38%	63.18%
F1	36.77%	29.45%	34.69%	68.70%

Table A.3. List of accuracy measurement scores across the pass (1) / fail (0) regex experiments.

Lexical Structure

Experiment 1: Word Overlap Measures

	Dice	**Jaccard**	**Cosine**	**Faucett**
Accuracy English	4.68%	2.59%	4.73%	7.66%
Accuracy German	2.54%	1.49%	2.79%	4.93%
Accuracy Spanish	3.73%	2.34%	3.63%	7.01%
F1 English	4.16%	2.85%	4.29%	5.38%
F1 German	2.17%	1.35%	2.35%	3.04%
F1 Spanish	3.40%	2.46%	3.42%	4.80%

Table A.4. English, German and Spanish accuracy and F1 measurements per word overlap method. Notice how the Spanish and German versions are slightly worse than the English version. This is likely do to several sources of noise. First, the translations from English to Spanish and English to German could have introduced noise. Second, the tools for stemming and preprocessing are, in general, less robust for languages other than English.

Experiment 2: N-Gram Comparisons

	MA1	**MA2**	**MA3**
R	0.355	0.421	0.449
Kappa	0.021	0.051	0.071
MAE	1.85	1.41	1.23

Table A.5. Correlation measures for multiple model answers.

	English	**German**	**Spanish**
Accuracy	24.23%	16.62%	23.13%
Precision	46.70%	47.46%	44.68%
Recall	24.23%	16.62%	23.13%
F1	29.25%	21.53%	27.76%

Table A.6. Comparative Scores for best-performing n-gram models across languages. Model Answers used are 3 and the n-gram size is 3 as well.

Semantics

Exp. 1: Knowledge and Corpus Based Meaning

Measure	**SP**	**LCH**	WP
Accuracy	5.97%	9.80%	13.1%
Precision	39.53%	32.48%	33.53%
Recall	5.97%	9.80%	13.1%
F1	4.95%	8.19%	12.50%

Table A.7. Accuracy Scores for Shortest Path (SP), Leacock & Chodorow (LCH) and Wu & Palmer (WP) conducted in the Spanish version of *WordNet*.

Exp. 2: Word Embeddings

Model	**Precision**	**Recall**	R
w2v	71.3%	74.3%	0.20
ft	71.9%	76.6%	0.17
w2v_ma2	74.3%	77.6%	0.24
ft_ma2	73.8%	77.2%	0.18
w2v_ma3	73.8%	77.3%	0.22
ft_ma3	74.4%	77.3%	0.18

Table A.8. Accuracy Scores for Pass/Fail binary classification. *Word2Vec (w2v)* and *FastText (ft)* models using $n = 3$ n-grams and model answers (ma) ranging from $1to3(ma3)$ are compared.

A.3 Algorithms

A.3.1 Regular Expression Algorithms

```
   Data: student_answers, question_regexes, partial_credit_weight
   Result: Numerical score ranging from [0, 1]
 1 begin
 2     hits ⟵ 0;
 3     total_patterns ⟵ LENGTH(question_regexes);
 4     foreach student_answer in student_answers do
 5         foreach regex in
             GET_FULL_CREDIT_REGEXES(question_regexes,
             student_answer) do
 6             if MATCH(regex, student_answer) then
 7                 hits ⟵ hits + 1.0;
 8             end
 9         end
10         score ⟵ (hits/total_patterns);
11         if score < 1.0 ∧
             PARTIAL_CREDIT_REGEXES_SIZE(student_answer) > 0
             then
12             foreach regex in
                 GET_PARTIAL_CREDIT_REGEXES(question_regexes,
                 student_answer) do
13                 if MATCH(regex, student_answer) then
14                     hits ⟵ hits + (1 ∗ partial_credit_weight);
15                 end
16             end
17             score ⟵ (hits/total_patterns);
18         end
19         return score
20     end
21 end
```

Algorithm 1: Regular Expression Matching Algorithm: it all student responses to a given question item. It iterates through each student responses and attempts to find a full credit match for each of the concepts the teacher has defined. If it fails to find a match and the teacher has defined partial credit regular expressions for the given concept, then the algorithm checks to see if the student response matches a partial credit regular expression and can receive partial credit for that concept. Finally, it averages the score by the total number of patterns i.e. concepts for the question which normalizes responses to a range of 0 (no matches) to 1 (all full credit matches).

A.4 Miscellaneous

A.4.1 Stembord Project Vision for the Future

The year is 2035, and Dave is about to graduate from high school at 16. There's nothing abnormal about this, about 15% of the students each year have already completed enough curricula requirements to begin studying at a university. Dave has been able to achieve this in large part because he has someone special. Every year since Dave was a child he has been tutored by Hal. Hal has a slightly stern personality, but he can be lovable and even funny at times. In fact, and Dave knows this, Hal has a personality adapted specifically for Dave. Its a personality designed to motivate Dave and encourage him, to keep him from slacking and give him a desire to achieve. Hal also knows exactly how to describe new materials in a way that clarifies for Dave how the subject works, if Hal had to describe the same topic to someone else he would likely use a somewhat different approach. Not only does Hal adapt his teaching strategies for Dave, but because Dave is dyslexic Hal presents materials in an accessible way optimized to alleviate dyslexic learning problems. Hal also has amazing visualization and clarification skills. Dave still remembers how Hal once took him on a VR journey in a 3D world where Dave could manipulate quadratic functions in real-time in order to model the shape, peak, and times of rockets' trajectories. Or how once for a literature class, Hal read him out loud The Canterbury Tales in the exact Middle English of the 14th century, he also created vocabulary lists for Dave of the Middle English words Dave didn't know, and after some training, took Dave into a simulation of the 14th century where Dave could speak to and question Chaucer's characters all in Middle English.

Hal even knows when Dave hasn't studied a piece of material in a while and gives him a helpful reminder and often impromptu customized quizzes which are designed to reinforce and ingrain in Dave's long-term memory subject matter that was covered perhaps months or even years before.

The breadth and depth of knowledge that Hal has makes him the perfect tutor. For Dave, he can have a one-on-one dialog with an expert in whatever subject he is studying from physics to anthropology, and because Hal knows so much and has a synthesis of all this knowledge - Hal is constantly consuming and synthesizing the research from every paper in every field each and every day - Dave can use Hal to help him when conducting research reports and doing statistical analyses. If Dave forgets which statistical test to run, Hal can not only tell him which is appropriate in a given scenario but why and also offer guided instruction so Dave doesn't need Hal's help as much in the future. Hal gives Dave, Dave's parents, and his teachers and school administrators formative and summative feedback about Dave's progress in a way designed to counter bias and not only increase Dave's own knowledge and skills but increase his teachers' and parents abilities to help him along his learning pathway.

Of course, Dave isn't the only one who has access to Hal, Hal's a rather democratic fellow and there's plenty of him to go around since he is after all, a program.

Everyone has access to Hal, from the oldest senior citizen in Sweden to the youngest preschooler in Venezuela. But the fact that he's so spread out or that he's a program doesn't bother Hal one bit. In fact, its been years since he has had to say, "I'm sorry Dave, I'm afraid I can't do that" in response to a question Dave or anyone else has posed.

B

Erklärung der Kandidatin / des Kandidaten

☐ Die Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen- und Hilfsmittel verwendet.

Namen der Mitverfasser: John Faucett

Datum	Unterschrift der Kandidatin / des Kandidaten