Spring 2010

An Automated Grader for Short Answer Responses

Sami Saqer
Macalester College

An Automated Grader for Short Answer Responses

Submitted to the Department of Mathematics, Statistics and Computer Science in partial fulfillment of the requirements for the degree of Bachelor of Arts

Sami Saqer

Advisor:	Prof. Daniel Kaplan
Second Reader:	Prof. Susan Fox
Third Reader:	Prof. Shilad Sen

MACALESTER COLLEGE

May 3, 2010

Abstract

Computers routinely grade multiple-choice questions by simply matching them to an answer key. Can they effectively score essay exams? This report examines an automated technique for grading short answer responses, using a grading system I have constructed. This system assigns a grade to a *student answer* based on its similarity to a *model answer* provided by an instructor. Similarity is measured using 1) the semantic similarity between isolated words, and 2) the similarity between the order of those words. The performance of the system was evaluated by scoring actual exam questions and comparing the computer-assigned grades to those given by human instructors. In favorable situations, the correlation between the computer- and the human-assigned grades was only a little less than between human instructors. Several characteristics of texts can cause the performance of the system to degrade. For example, the system's performance is poor with answers that contain negations, truly unique phrases or idioms, misspelled words, or contradictory information.

Acknowledgments

I would like to thank my advisor Professor Kaplan for his guidance and support throughout this project. I would also like to thank Professor Alicia Johnson for her help in grading some of the answers in the dataset, both my readers Professor Susan Fox and Professor Shilad Sen for reading my paper and giving me technical support.

Contents

Chapter 1

Introduction

1.1 Introduction & Motivation

The classic Bloom's Taxonomy arranges learning objectives as a pyramid, with knowing the facts at the bottom, tapering up to comprehension, application, analysis, synthesis, and, at the top, evaluation (Bloom, 1969). A learner needs to attain knowledge and skills at the lower levels before attaining them at the higher levels. Therefore, the assessment of the knowledge acquired by the learner is crucial to the learning process. Multiple choice and true/false questions are appropriate at the lower levels of Bloom's Taxonomy since factual knowledge and understanding of vocabulary can be tested in this format in a straightforward way. Additionally, multiple choice questions can be automatically graded without sophisticated text understanding (i.e by simply matching to an answer key). Multiple choice questions, however, might not honestly assess students' higher order cognitive skills. They assess only student's ability to choose an answer correctly from a list of possible answers, rather than to freely construct one.

Short answer questions, on the other hand, require students to formulate their own free text response. They reinforce learning at the lower levels of Bloom's taxonomy and help develop higher cognitive skills, as they assess understanding without offering clues or plausible choices. However, the overhead and logistics of collecting and grading short answer responses and, more importantly, the lack of a fully automated system to handle these tasks, made it harder to use the them, specially in large-scale educational testing. An automatic grading system could potentially reduce the difficulties of assessment and decrease grading errors by providing a "double check" for grades. It would also be consistent in the way it grades, offering more objectivity in the grading process.

Researchers do not exactly know how humans grade text answers, but common sense suggests that an instructor grades a text answer by determining the meaning of the answer. Unfortunately, meaning itself is an abstract idea, requiring sophisticated analysis of the answer text in order to understand its meaning. Despite the many advances in the field of natural language processing, full comprehension of natural language text remains beyond the power of machines. The inherent complexity and flexibility of natural languages poses a challenge to implementing such a system. For example, words have multiple meanings; a sentence can be paraphrased in many different ways and retain the same meaning; the meaning of a word, a sentence or phrase can change depending on the context surrounding it; a small change in wording can drastically alter meaning.

Recent advances in natural language processing techniques allow us to create systems that automatically grade free text responses without having to fully understand answers. One such technique assigns grades by measuring the semantic similarity between short texts. To evaluate the possibilities and pitfalls of this approach, I have built a system that examines the similarity in isolated word meaning and word order between a short essay and a model answer. In this paper, I present my approach in detail and the results of experiments evaluating the performance of the system in comparison to human graders.

1.2 Background

For the purposes of this project, I define a short answer question as a question which warrants a response from a few phrases up to two sentences. It has a specific set of answers that the examiner is looking for, and often has an objective criterion for correct and incorrect answers. Consider the question: *What is the difference between a variable and a "model term"?*

Such a question is constrained to a limited response; hence, it is appropriate to automate its grading. On the other hand, consider the following open-ended question: *"Compare and contrast the course of industrialization in western Europe v.s Russia"*. Such a question might be answers in a variety of ways not anticipated by the instructors. Additionally, unlike a full essay in which style and grammar may be important components of the grade, the grade assigned to each response is primarily based on content that indicates knowledge and understanding.

An automatic short answer grading system bears similarity to human graders. Both of them have a grading schema of some form containing acceptable answers for each question, which we will call *model answers*. Both of them compare a *candidate answer* to the *model answer(s)* and assign a grade based on how closely the candidate answer matches the model answer. An automatic grading system should recognize an answer as correct when it is a paraphrasing of the model answer. Paraphrases may be due to different syntax, different inflections of a word, or substitution of synonyms.

Consider, for example, the question *What causes day and night?*

Model Answer	The Earth spins on its axis.
Answer1	The result of the rotation of our planet around itself.
Answer2	Clouds block out the sun's light.

Table 1.1: *Examples of short answer responses*

A grading system should recognize that answer 1 is a correct answer despite having very few words in common, as it is a paraphrasing of the model answer containing different synonyms. It should also determine that answer 2 is incorrect, as it has a completely different meaning from the model answer.

One traditional automated approach to this problem is keyword analysis, which analyzes the text by looking for the presence or absence of predetermined keywords, but it would simply fail to recognize correct answers that do not contain any of predetermined key words, even if the answer contains synonyms or different inflections of the key words.

1.3 Aim

In this project, I aim to develop an automated system for grading free-text responses for short answer questions. By an automated system, I mean a system that does not require the grader's manual work beyond constructing one or more model answers. I frame this problem as a text similarity task. I develop a system that assigns a grade to a *student answer* based on its similarity to a *model answer* provided by an instructor. I attempt to measure similarity using 1) the semantic similarity between isolated words, and 2) the similarity between the order of those words. My objectives are to evaluate the extent to which this approach can reproduce human instructor grades and to examine what features of answers support and undermine the ability to grade automatically.

The next section gives an overview of previous work on text similarity and automatic grading. Chapter 3 describes my approach to tackle this problem, a walk-through example and some implementation details. Chapter 4 provides the results of brief experiments and an analysis of the correlations with the human grader in order to evaluate the performance of the system. Chapter 5 presents my conclusions and proposes future extensions of this work.

1.4 Related Work

Several approaches have been suggested for the general task of measuring the similarity between documents or texts and others specifically for automatic grading of essays and short answer questions. Previous work particularly relevant to my task falls into three rough categories:

- Information Extraction Approaches
- Corpus-Based Approaches
- Machine Learning Approaches

Below, I give a quick overview of the most relevant work from each of the above categories in order to explore their strengths and limitations and identify the challenges of automatic short answer grading.

1.4.1 Information Extraction Approaches

Information extraction is a natural language processing technique that turns the unstructured information embedded in texts into structured data (Jurafsky and Martin, 2008). For the automatic grading task, information extraction approaches are conceptually similar to my approach. All of them attempt to match a student response to a model answer. Most information extraction approaches require manually-crafted patterns, which if matched, indicate that a question has been answered correctly. A pattern represents all the paraphrases of the correct answer collapsed into one. Intelligent Assessment Technologies developed a system called Automark (Mitchell et al., 2002) that represents the content of the grading scheme as syntactic-semantic templates. Student answers are first parsed and then intelligently matched against the grading scheme template. Another IE approach by Sukkarieh et al. (2003) uses information extraction techniques to extract significant features from answers and match them with hand crafted patterns. This approach requires skills, labor, and familiarity with both domain and tools. A different system by Callear et al. (2001) compares model answers with a student's answer by parsing sentences into concept dependency groups. Matching is carried out between the concepts and dependencies found in students' answers and those found in the model answers. They claim to deal with grammatically incorrect and ambiguous answers. This approach is programming intensive, and incomprehensible to normal instructors, and therefore was not useful for this project.

1.4.2 Corpus-Based Approaches

Corpus-based approaches are based on statistical information of words in a huge corpus (a collection of written texts). A well known corpus-based approach is the Intelligent Essay Assessor (IEA) (Foltz et al., 1999). It analyzes the textual context using Latent Semantic Analysis (LSA) (Landauer et al., 1998). LSA is a technique primarily used for information retrieval tasks. It is based on word-document co-occurrence statistics (such as TF-IDF[1]) in the training corpus represented as a matrix. Each document is

[1]Document Frequency-Inverse Document Frequency

represented using its words as a vector in n-dimensional space, where *n* is the number of words in the document. The matrix is decomposed by singular value decomposition (SVD) into the product of three other matrices. The dimension of the diagonal matrix is reduced by eliminating the small singular values. The original matrix is then reconstructed from the reduced dimensional space. The similarity between two documents is measured by computing the similarity between the vectors representing for each the document in the reduced dimension space.

In IEA, LSA is first trained on domain-specific corpus. Based on this training, LSA derives a matrix representation of the information contained in the domain. A student essay is then represented by an LSA vector based on the combination of its words. This vector can then be compared with vectors for essays in corpus. The angle between the two vectors represents the degree of their similarity (Foltz et al., 1999). LSA evaluates a text based on its choice of words. It does not take into account any structural information and requires a large corpora as basis for comparison. Additionally, it relies on the assumption that more similar essays share more of the same words. However, in short essays, word co-occurrence may be rare. Thus the LSA is more appropriate for larger texts.

Kanejiya et al. (2003) developed a Syntactically Enhanced LSA (SELSA) system, which generalizes LSA by considering a word along with its syntactic neighborhood given by the part-of-speech tag of its preceding word, in order to add context to LSA. They concluded that LSA is slightly better than SELSA in terms of the correlation with human graders. But SELSA is at least as good as LSA in terms of the mean absolute difference measure.

The E-rater system (Burstein et al., 2001) is another corpus-based approach, designed to produce holistic scores for essays based on the linguistic features of effective writing that faculty readers typically use: organization, sentence structure, and content. It uses actual essays scored by instructors to predict the holistic score a grader would give to an essay. The system uses shallow parsing techniques to identify syntactic and discourse features. It does not assess text content at the formative level and, therefore, is not suitable for the short answer questions.

C-rater is another similar system developed by Leacock and Chodorow (2003) at the Educational Testing Service (ETS). It is based on analyzing the logical relations between the syntactic components for each sentence in the answer. It tries to match the syntactical features of a student response (subject, object, and verb) to that of a set of correct responses. It takes word order into account. Again, this approach is inappropriate for short answer essays as it doess not assess essays at the formative level.

1.4.3 Machine Learning Approaches

As mentioned before, information extraction approaches require manually crafted patterns. Constructing these patterns, however, is a laborious process that needs expertise both in the domain of the examination, and in computational linguistics. To save time and labor, various researchers have investigated machine learning approaches to learn IE patterns and automate the writing process. For example, Sukkarieh et al. (2004) try to improve and refine their earlier system (Sukkarieh et al., 2003) using machine learning techniques such as the Nearest Neighbor Classification techniques. They require a corpus to be annotated, indicating which sentences in a text contain the relevant information for particular pattern. Then more patterns can be learned by bootstrapping those annotated patterns. Pulman and Sukkarieh (2005) compare several machine learning techniques, including inductive logic programming, decision tree learning, and Bayesian learning to the earlier information extraction techniques. They conclude that machine learning methods are not accurate enough to replace the hand-crafted, pattern-matching approach.

Hatzivassiloglou et al. (1999) propose an approach for the more general, but related task of text similarity. In this approach, a sentence is represented in a vector consisting of values of primary features, such as word co-occurrence, matching noun phrases, and composite features, such as a pair of nouns appear within 5 words from one another in both texts. Similarity between two texts is determined using a classifier trained over a corpus of manually marked pairs of units texts. The drawback of this approach is that the preparation of a training vector set could be an impractical, and time-consuming task. Therefore, it is inappropriate the automatic short answer grading task.

1.4.4 Semantic Text Similarity

The approaches I listed above have several weaknesses. First, some of them require the grader's manual work for preprocessing the text or constructing answer patterns. Second, some of them require the availability of large training data. Specifically some approaches would require large numbers of graded answers to the particular question the instructor wants to ask. Hence, they cannot be easily adapted to other domains. Third, some methods compute similarity according to the co-occurring words in the texts. They work well for long texts because they have adequate information for manipulation by a computational method. But are much less effective with shorter texts.

To address these issues, I tackle this problem from an arguably simpler approach. My approach does not require sophisticated analysis and understanding of the text in the answer, training data or grader's manual work, beyond providing a *model answer*. I frame the problem of automatic grading as a text similarity task. A candidate answer is assigned a grade based on its "similarity" to a *model answer* provided by an expert in the field (e.g instructor). The similarity between the two pieces of text is measured as a function of 1) the semantic similarity between isolated words, and 2) the similarity of the sentence structure, approximated by word order. The next chapter provides details of my approach.

Chapter 2

A Text Similarity Approach

2.1 Overview of the system

I frame the problem of automatic short answer grading as a text similarity task. Two texts are similar if they convey the same information. Thus, my approach is to determine the similarity between the candidate answer and the model answer at the semantic level. The meaning of a sentence is conveyed by its component words and their structure (i.e order). Therefore, my approach is to derive text similarity from semantic and structural information contained in the compared sentences. I use two similarity functions and combine them together to obtain a single score: 1) Word-to-word semantic similarity, and 2) word-order similarity that incorporates syntactic information. Figure 2.1 shows the procedure for computing the text similarity between two texts. Compared texts progress through a number of preprocessing steps before measuring the text similarity and producing the final score. These stages include tokenization, part-of-speech tagging, normalization, and lemmatization. I will go through all of these in detail in the next section.

After the preprocessing stages, I end up with two list of words representing each answer. Then, using a word-to-word semantic similarity function, I derived text-to-text semantic similarity. In order to incorporate structural information, I measure the similarity of the order of the matched words in the two texts. I form a word-order vector for each text with regard to the other text, and then use the a similarity metric as an indicator of how close the word order is to the answer. Finally, the overall text similarity is derived by combining semantic similarity and word-order similarity into a single score.

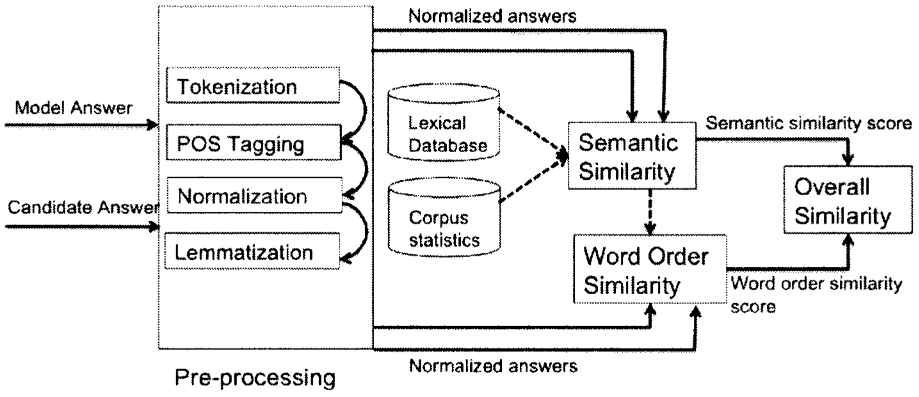

Figure 2.1: *Overview of the System's Structure*

The following sections present a detailed description of each of the above steps.

2.2 Preprocessing

Before measuring text similarity, the incoming text is preprocessed to eliminate any extra noise, and focus on the essence of the text. Consider for example the following sentence obtained from the dataset used in this study as described in section 3.1, which I will use throughout this section to illustrate each individual preprocessing step.

> Interaction terms included when the role of an explanatory variable is modulated by another explanatory variable

2.2.1 Tokenization

Tokenization is the process of breaking down a string into identifiable linguistic units that constitute a piece of language data.

First, if the answer text consists of more than one sentence, it is broken down into separate sentences. Then each sentence is tokenized into words. Note that tokenizing a string by splitting on whitespace is not sufficient, as it does not eliminate other non-alphabetical characters attached to words such as parenthesis or punctuations. For example "(*word,*" has to be split by the comma and the parenthesis.

When the string is tokenized, we obtain a list of words representing the full text in their original order.

```
[Interaction,
terms,
included,
when,
the,
role,
of,
an,
explanatory,
variable,
is,
modulated,
by,
another,
explanatory,
variable]
```

2.2.2 Part-of-Speech Tagging

Part-of-speech (POS) tagging is the process of classifying words into their parts-of speech (nouns, verbs, adjective, etc) and labeling them accordingly. A well known technique for POS tagging, the Hidden Markov Models, is a statistical model in which the system being modeled is assumed to be a Markov process with unobserved state (Jurafsky and Martin, 2008). The HHM tagger chooses the most likely POS-tag sequence. that maximize the product of word likelihood and tag sequence probability.

Applying this step results in:

> (Interaction, NN), (terms, NNS), (included, VBD), (when, WRB), (the, DT), (role, NN), (of, IN), (an, DT), (explanatory, NN), (variable, JJ), (is, VBZ), (modulated, VBN), (by, IN), (another, DT), (explanatory, NN), (variable, JJ)

The labels next to each word identifies the POS tags. For instance *NN* for singular nouns, *NNS* for plural nouns, *VBD* for past tense verbs, *WRB* for adverbs, etc.

2.2.3 Normalization

Normalization involves eliminating all special characters, punctuations, capitalization, and stop words (such as prepositions, pronouns and help verbs e.g can, will, may), as they contribute less to the meaning of a sentence than other words. Applying this process to the example,

> [(interaction, NN), (terms, NNS), (included, VBD), ~~(when, WRB)~~, ~~(the, DT)~~, (role, NN), ~~(of, IN)~~, ~~(an, DT)~~, (explanatory, NN), (variable, JJ), ~~(is, VBZ)~~, (modulated, VBN), ~~(by, IN)~~, (another, DT), (explanatory, NN), (variable, JJ)]

The normalized output is

> [(interaction , NN), (terms , NNS), (included , VBD), (role , NN), (explanatory , NN), (variable , JJ), (modulated , VBN), (another , DT), (explanatory , NN), (variable , JJ)]

2.2.4 Lemmatization

Lemmatization is the process of removing affixes of words to reduce the word to its base, ensuring that the resulting word appears in the dictionary. For example, the verb 'went' lemmatizes to 'go' and 'women' lemmatizes to 'woman'.

> [(interact , NN), (term , NNS), (include , VBD), (role , NN), (explanatory , NN), (variable , JJ), (modulate , VBN), (another , DT), (explanatory , NN), (variable , JJ)]

2.3 Word-to-Word Semantic Similarity

My model of the semantic similarity of two texts first includes a function of the semantic similarity of their component words. There is extensive literature on word-to-word similarity metrics. These range from distance-oriented measures computed on semantic networks, such as the WordNet, to statistical metrics learned from large text collections (Budanitsky and Hirst, 2001). Semantic networks organize concepts into a hierarchical tree of concepts (see section 2.8.1 for more details on semantic networks).

In distance-oriented metrics, which are based semantic networks, the similarity between two concepts in the hierarchy can be measured as a function of the shortest path connecting them in that space and/or the their depth in the taxonomy (Wu and Palmer, 1994; Leacock et al., 1998).

Some statistical metrics use corpora of web content obtained from web search engines. For example, Turney (2001) defined a Pointwise mutual information (PMI-IR) measure, using the number of hits returned by a web search engine, to recognize synonyms. Bollegala et al. (2007) propose a method which integrates both the number of hits returned for a query and lexico-syntactic patterns extracted from snippets to measure semantic similarity between a given pair of words. The number of hits for the query *P* AND *Q* can be considered as a global measure of co-occurrence of words P and Q, while snippets provide useful information regarding the local context of the query term.

A number of hybrid approaches have been proposed that combine a knowledge-based lexical taxonomy, with information content from corpus statistics (Resnik, 1995; Jiang and Conrath, 1997; Lin, 1998). For example, Jiang and Conrath (1997) measure the information content of two concepts by computing their least common subsumer (LCS) in WordNet. Information content is a measure of the specificity of a concept, and the LCS of two concepts is the most specific concept that is an ancestor of both concepts in the semantic hierarchy. Budanitsky and Hirst (2001) provide an extensive comparison of these approaches and conclude that Jiang and Conrath (1997)'s approach performed best.

Li et al. (2003) suggest a new approach which outperformed all the previous approaches. They propose that the similarity between two words $sim(w_1, w_2)$ be a nonlinear function of the shortest path length between the two words (l) and depth in of the least common subsumer in the semantic hierarchical (h). The nonlinear function keeps the interval of similarity between [0, 1].

When the path length decreases to zero, the similarity monotonically increases toward the limit 1; when the depth of the subsumer increases toward infinity, the similarity monotonically increases to the limit 1. They propose the following formula for similarity:

$$sim(w_1, w_2) = e^{-\alpha l} \cdot \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} \tag{2.1}$$

where $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ are parameters scaling the contribution of shortest path length and depth, respectively. The optimal values of α and β are dependent on the knowledge base used and can be determined using a set of word pairs with human similarity ratings. They report that the optimal parameters for WordNet are: $\alpha = 0.2$ and $\beta = 0.6$.

In this project I use the metric proposed by Li et al. (2003) as they reported the highest correlation with human judges comparing to all the other word similarity metrics. I also follow Li et al. (2003)'s example of drawing upon the WordNet (Miller, 1995) as the semantic knowledge-base.

2.4 Sentence-to-Sentence Semantic Similarity

A sentence is composed of words, ordered in a particular way. In the previous section, I define a metric that determines the semantic similarity between individual words. Using this metric, I measure the *semantic* similarity between sentence A and sentence B in the following way. First both sentences are preprocessed as described is section 2.2. For each word in sentence A, I pick the word in sentence B that has the maximum semantic similarity to that word in A, according the word similarity metric.

Each word has an information content, which I use to scale its contribution to the sentence's semantic similarity. Information content is a measure of specificity of a word, derived from its probability in a corpus. Words that occur with a higher frequency in a corpus contain less information than those that occur with lower frequencies (Resnik, 1995). Specific words have more content, hence contributing more to the meaning of the whole sentence. Based on this intuition, I weight a word using the information content derived from the Brown corpus. By doing so, I can give a higher weight to a semantic matching identified between two specific words (car, motorcar) and give less importance to the similarity measured between generic concepts (entity, object).

The similarity scores are multiplied by their respective information content. The results are then summed up, and normalized by the sum of all individual information content scores.

The same process is repeated to determine the most similar words in sentence A starting with words in sentence B. Finally the resulting similarity scores are combined using a simple average of the similarity score of each sentence with respect to the other.

To formalize this process, consider the following two sentences:

$$A = [w_1, \ldots, w_i, \ldots, w_m], B = [v_1, \ldots, v_j, \ldots, v_n]$$

I construct an $m \times n$ semantic similarity matrix, where m is the number of words in sentence A, and n is the number of words in sentence B, after they have been preprocessed.

$$Sim\_Matrix = \begin{bmatrix} \sigma_{1,1} & \cdots & \sigma_{1,j} & \cdots & \sigma_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_{i,1} & \cdots & \sigma_{i,j} & \cdots & \sigma_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \sigma_{m,1} & \cdots & \sigma_{m,j} & \cdots & \sigma_{m,n} \end{bmatrix}$$

The $\sigma_{i,j}$ entry in the matrix is the semantic similarity score computed between w_i in A and w_j in B using the method presented in section 2.3, for all $i = 1...m$ and $j = 1...n$. More specifically, the score is determined in the following way:

- If w_i is the same word as v_j, then the score is set to 1.0, indicating perfect match.
- If both words are tagged as proper nouns, the score is determined using a string matching algorithm which determines the similarity of two strings based on the longest common sequence. The reason I used string similarity here is that semantic similarity based on the WordNet can not provide any similarity value between proper nouns, because they do not exist in the dictionary. Also, if one proper noun is slightly misspelled, the string similarity metric will still be able to recognize it and a give a fairer score.
- If the score obtained from the semantic similarity metric is less than a preset threshold, then score is set to 0.
- Otherwise, the score is set to the same score obtained from the semantic similarity metric.

I use a threshold because that the similarity scores may be very low, indicating that the words are highly dissimilar. In this case, I would not want to introduce such noise to the metric.

After the similarity matrix is constructed, for each word w_i in A, I pick a word v_j in B that has the highest semantic similarity score. The similarity score is then weighted with w_i's information content. Next, the same process is applied for each v_j in B. These weighted similarities are then summed up and normalized with the sum of the information contents. Finally the resulting similarity scores are combined using a simple average. The formula below summarized the semantic similarity function between the two sentences A and B.

$$semanticSim(A, B) = \frac{1}{2}\frac{\sum_{i=0}^{m} Sim(w_i, B) * ic(w_i)}{\sum_{i=0}^{m} ic(w_i)} + \frac{\sum_{j=0}^{n} Sim(v_j, A) * ic(v_j)}{\sum_{j=0}^{n} ic(v_j)} \tag{2.2}$$

where $Sim(w, X)$ is the highest semantic similarity between a word w in another in the text X. The term ic is a function for obtaining the information content of the given word.

This semantic similarity score has a value between 0 and 1, with a score of 1 indicating identical sentences, and a score of 0 indicating no semantic overlap between the two sentences.

2.5 Word Order Similarity in Sentences

In order to incorporate the basic structural information carried by a sentence, I take into account the similarity in word order in the two sentences. The idea here is to compare the order of words in each matching word-pair. Each word in A is assigned an index from 1 to m, based on the position of that word in A. The process is repeated for B. Based on matching word-pairs (determined in the previous section), I create two word-order vectors. The first vector simply represents the words in A. The second vector's components are the positions in B of the words that pair with those in A. Consider, for example, the following matches have been determined between sentence A and sentence B:

Sentence A	Sentence B
w_1	v_1
w_2	v_3
w_3	v_5
w_4	v_7

Then I end up with vectors $r_1 = \begin{bmatrix}1\\2\\3\\4\end{bmatrix}$ and $r_2 = \begin{bmatrix}1\\3\\5\\7\end{bmatrix}$ representing the word order of the two sentences.

Then to compute the similarity between the two word order vectors I used the cosine similarity, which is defined in the following way:

$$OrderSim = \frac{r_1 \cdot r_2}{\|r_1\|\|r_2\|}$$

The cosine similarity metric basically calculates the geometric angle between these two vectors, which corresponds to score of similarity between the two samples. The resulting similarity ranges from 0 meaning completely different order, to +1 meaning exactly the same order, and in-between values indicating intermediate similarity or dissimilarity.

The overall word order similarity score combines the word order similarity of each sentence in turn with respect to the other in a simple average. That is, I first compute the word order score for words from A matched to words from B, and the other way around, words from B matched to words from A. The two scores are then averaged.

Intuitively, structural information carried by the sentence contributes less to the overall meaning conveyed by the sentence. When determining the overall similarity between two sentences, then, word order similarity will be weighted less compared to the semantic similarity.

2.6 Overall Measure of Similarity

The main goal in this study is to automatically compute a score between 0 and 1 that indicates the similarity between the model answer M and the candidate answer C at the semantic level for the automatic grading task. Semantic similarity represents the lexical similarity. Word order similarity provides information about the relationship between words: which words appear in the sentence and which words come before or after other words. Both semantic and structural information (in terms of word order) play a role in conveying the meaning of sentences. Thus, the overall sentence similarity is defined as a combination of semantic similarity and word order similarity:

$$Sim(M,C) = (\alpha)SemanticSim(M,C) + (1-\alpha)OrderSim(M,C) \quad (2.3)$$

where α decides the relative contributions of semantic and word order information to the overall similarity computation. Since the structure plays a less important role for semantic processing of text, $(1-\alpha)$ should be a value less than 0.5.

An answer text may consist of multiple sentences. In that case, the answer text is first broken down into its sentence components. Then for each sentence in the model answer, I find the maximum similarity score that can be obtained by evaluating it against each sentence in the candidate answer using equation 2.3. The final similarity score is then computed by taking the average of all maximum sentence similarities.

Overall, the similarity method requires two parameters to be determined before use: a threshold for word semantic similarity (below which the similarity score is set to 0) and a factor α for weighting the significance between semantic information and structural information. Since syntax (approximated by word order) plays a less important role for semantic processing of text, I weight the semantic part higher, **0.85** for α. For the word similarity threshold, I need a threshold that eliminates any unnecessary noise and, at the same time, captures the semantic similarity between words. If the threshold is too small, it will introduce a lot of noise. If it is too large, it may not capture some important similarities. Both cases will worsen the performance of the system. After several empirical experiments I found that **0.40** to be a good threshold for the word-to-word semantic similarity.

2.7 A Walk-Through Example

In this section I illustrate how the system works as a whole by walking through a simple example. Given a model answer and a candidate answer for a question, as shown below, the system determines a score for the candidate answer that reflects its similarity to the model answer. Consider for example the question "What does a function signature in Java include?" A model answer M and a candidate answer C for this question are:

M = *"The name of the function and the types of the parameters"*
C = *"It includes the name of the method and the types of its arguments"*.

Note that answers differ in both details that are proponent to meaning (e.g. "of the" v.s "of its") and in using synonym pairs- e.g ("method/function" and "parameter/argument").

First, the two texts are preprocessed. This step includes tokenizing the text, tagging parts-of-speech , eliminating all special characters and punctuations, removing all stop words and lemmatizing the rest of the words.

After the preprocessing we end up with the following list of words. Note that the order of the words is retained:

M = [(name , NN), (function , NN), (type , NNS), (parameter , NNS)]
C = [(includes , VBZ), (name , NN), (method , NN), (type , NNS), (argument , NNS)]

Once the texts are preprocessed, an $m \times n$ semantic similarity matrix is constructed, where m is the number of words in M and n is the number of words in C, using the word-to-word semantic similarity metric described in section 3.3. The resulting matrix is shown below. As mentioned before, I used 0.40 as a threshold for word-to-word semantic similarity. In other words, if the similarity score obtained from the metric is < 0.40, the score is converted to 0 to eliminate unnecessary noise. Consider for example the word pair *type* and *function*. The similarity score obtained from the metric is 0.18, which is < 0.40, and therefore, the score is converted to 0. On the other hand, the similarity score between *parameter* and *argument* is 1.0, so it is returned. Note that if Wordnet were made specific for computer science language uses, then*function* and *method* would have a much higher similarity

$$\begin{array}{lccccc}
 & includes & name & method & type & argument \\
name & 0.0 & 1.0 & 0.0 & .55 & 0.0 \\
function & 0.0 & 0.0 & 0.47 & 0.0 & 0.0 \\
type & 0.0 & 0.63 & 0.0 & 1.0 & 0.0 \\
parameter & 0.0 & 0.0 & 0.0 & 0.0 & 1.0
\end{array}$$

After the matrix is constructed, for each row, the column with the highest value is selected. Then the score is weighted using the information content of the row's word. After that the information content scores (measured from the probability of encountering an instance of concept c in a large corpus) and the weighted similarity scores are summed together. Table 2.1 illustrates these steps.

The same process is repeated but this time starting with words in each column representing words in the candidate answer.Table 2.2 summarizes these results.

Using equation 2.2, the overall semantic similarity score is computed. As shown below:

$$SemSim = \left(\frac{1}{2}\right)\frac{30.41}{34.99} + \frac{29.59}{41.21} = 0.85$$

The next step is to compute word-order similarity between the two sentences in order to incorporate basic structural information. The system then assigns each word a word-order based on its position in the lists. Each word in M is assigned an order value from 1 to m, based on the position of that word in M. For example, the word *type* is assigned 3 and the word *parameter* is assigned 4. The same process is done for C. So the word *name* is assigned number 2, etc. Based on the word-to-word semantic similarity results from table 2.1, the system creates two word-order vectors. The first vector simply represents the words in M. The second vector's components are the positions in C of the words that pair with those in M. This is illustrated in the table below:

Word 1	Order	Word 2	Order
name	1	name	2
function	2	method	3
type	3	type	4
parameter	4	argument	5

So we end up with vectors $r_1 = \begin{bmatrix}1\\2\\3\\4\end{bmatrix}$ and $r_2 = \begin{bmatrix}2\\3\\4\\5\end{bmatrix}$ representing the word order of the two sentences.

Word 1	Word 2	SimScore	Information Content	Weighted Score
name	name	1.0	8.50	8.50
function	method	0.47	8.65	4.07
type	type	1.0	7.91	7.91
parameter	argument	1.0	9.93	9.93
		Sum	**34.99**	**30.41**

Table 2.1: *Word-to-word semantic similarity scores weighted with information content for rows representing words in the model answer.*

Word 1	Word 2	SimScore	Information Content	Weighted Score
includes	parameter	0.0	1.0	0.0
name	name	1.0	8.50	8.50
method	function	0.47	6.77	2.46
type	type	1.0	7.91	7.91
argument	parameter	1.0	9.93	9.93
		Sum	**35.11**	**29.52**

Table 2.2: *Word-to-word semantic similarity scores weighted with information content for columns representing words in the candidate answer*

I then use Pearson's correlation to compute the similarity between the two vectors, which in this case is $OrderSim_1 = 1.0$

Again, the same process is repeated but now using the word-to-word semantic similarity results from table 2.2 as shown below.

Word 1	order	Word 2	Order
name	2	name	1
method	3	function	2
type	4	type	3
argument	5	parameter	4

We obtain the vectors $r_1 = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \end{bmatrix}$ and $r_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$, and their similarity score is $OrderSim_2 = 1.0$.

The overall word order similarity score is the average of these two scores, which is $OrderSim = 1.0$

Finally, the overall sentence similarity is defined as a combination of the semantic similarity and word order similarity. Since semantic similarity plays a more important role than word order similarity, it is weighted more heavily. Using Eq. 2.3, the overall similarity score is

$$(\alpha)SemSim + (1 - \alpha)OrderSim$$

where $\alpha = 0.85$, which decides the relative contributions of the semantic similarity and word order similarity to the overall all similarity. Therefore the overall similarity score is 0.87.

2.8 Implementation Details

I implemented the system with Python because it is a simple yet powerful programming language with excellent libraries for natural language processing. I also use a third party library called Natural Language Toolkit (NLTK) (Bird et al., 2009) that supplies basic classes and standard interfaces for performing the preprocessing tasks such as tokenization, part-of-speech tagging and lemmatization. It also provides an efficient interface to access the WordNet corpus.

2.8.1 WordNet Lexical Database

For the word-to-word similarity metric, I use the WordNet lexical database. WordNet is a manually-constructed lexical system developed by George Miller et al. (1990) at the Cognitive Science Laboratory at Princeton University. Originating from a project whose goal was to produce a dictionary that could be searched conceptually instead of only alphabetically, WordNet evolved into a system that reflects current psycholinguistic theories about how humans organize their lexical memories.

The basic object in WordNet is a set of strict synonyms called a synset. By definition, each synset in which a word appears is a different sense of that word. WordNet consists of four main divisions, one each for nouns, verbs, adjectives, and adverbs. WordNet 3 (2006), the version of WordNet used in this study, contains $117,659$ synonym sets and $20,6941$ senses in all of the 4 divisions. Because synsets contain only strict synonyms, the majority of synsets are quite small. Similarly, the average number of senses per word is close to one. Figure 2.2 illustrates a fragment of the semantic hierarchy of WordNet.

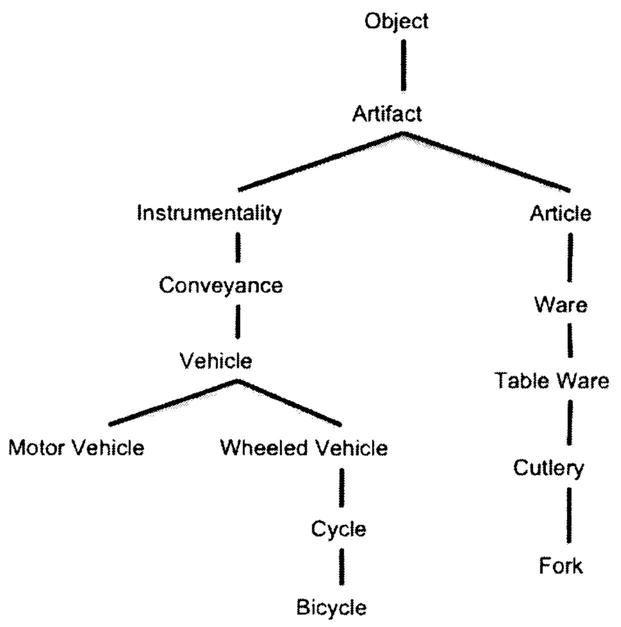

Figure 2.2: *An example of WordNet Lexical Database, from most abstract to most specific*

2.8.2 Information Content

As discussed in section 2.4, information content is a measure of specificity/significance of a word. The value of the information content of a word is derived by estimating the probability of occurrence of this word in a large text corpus. Hence, words that occur with a higher frequency in a corpus contain less information than those that occur with lower frequencies (Resnik, 1995). Following this notation, Jiang and Conrath (1997) suggested the following formula to quantify the information content (IC) of a word:

$$IC(w) = logP^{-1}(C) \tag{2.4}$$

where P(w) is the probability of encountering an instance of a word w. NLTK provides a utility, called `WordNetICCorpusReader` , which reads in a file of a precomputed probabilities of all word in the WordNet based on their occurrence in a given corpus. In this case, I used the Brown corpus. Below I provide more detail about it.

Algorithm 1 Pseudo-code to computer information content of a word

```
Input: corpus C, word w
Output: The information content of w based on C corpus statistics.
$corpusStats \leftarrow WordNetICCorpusReader(C)$
$P = corpusStats[w]$
$ic \leftarrow -log(P)$
return $ic$
```

Brown Corpus

The Brown Corpus (Francis and Kucera, 1979) of Standard American English was the first of the modern, computer readable, general corpora. Compiled by Francis and Kucera of Brown University, it was the first million-word electric corpus of English. The corpus contains text from 500 sources. Sources were sampled from 15 different text categories, such as news, editorial, government, and so on, which provided a wide variety of vocabulary.

2.8.3 Program Structure

The program has 4 main classes:

- **Word**: represents an individual word in the text. It has attributes such as the actual string representation of the word, its part-of-speech tag, sense (meaning) and specificity (its information content). It also contains the logic measuring the semantic similarity between this word and another using the method described in section 3.3 and the string similarity.
- **Response**: represents the candidate answer and the model answer. A *response* object is made up of *word* objects representing the individual words in the answer text. The response class also has attributes such as the string representation of the answer text, grades assigned by humans, grades assigned by the system and problem ID it belongs to. As well it contains the logic for the preprocessing step and computing overall similarity with another response object.

- **Utils**: contains mainly utility functions for reading-in and parsing input files.
- **GradingEngine**: puts everything together. It has the logic for automatically grading a list of answers by calling the text similarity function in the *response* class. It also has a function for evaluating the performance of the program, such as computing the Pearson correlation efficient.

2.8.4 Semantic Word Similarity Function

The function `semanticSimilarity()` in the *word* class measures the semantic similarity between this word and a given word. It returns a score between $[0, 1]$. As discussed in section 3.3, it uses Li et al. (2003) word similarity function with WordNet Lexical database. Each word (sense) in the WordNet has one or more synonyms. This function find the maximum semantic similarity score that can be obtained by pairing up each synonym from the first with another synonym from the other word. Algorithm 3 shows the pseudo-code for this function.

Algorithm 2 Pseudo-code for the semantic word similarity function

```
**Input**: word1, word2
**Output**: Semantic score between Word1 and Word2 between [0,1]
$highestSim \leftarrow 0$
**for all** $syn1$ in word1.synonyms **do** {//synonyms from the WordNet}
  $i \leftarrow i + 1$
  **for all** $syn2$ in word2.synonyms **do**
    $similarity \leftarrow Li\_Similarity(syn1, syn2)$
    **if** $similarity > highestSim$ **then**
      $highestSim \leftarrow similarity$
    **end if**
  **end for**
**end for**
**return** $highestSim$
```

2.8.5 String Similarity Function

The function `stringSimilarity()` in the *word* class measures the "similarity" of two word strings. It is used to determine the similarity between proper nouns identified in the text. It returns a score between $[0, 1]$, where as a rule of thumb, a value over **0.6** means the strings are close match. To implement this function, I used the built-in string similarity function in Python called `sequenceMatcher`, which finds the longest contiguous matching subsequence. The same idea is then applied recursively to the pieces of the sequences to the left and to the right of the matching subsequence. The algorithm then computes the sequences similarity score by dividing the $2\times$ the total number of matches found by the total number of characters in both sequence. More formally, $\frac{2*S}{L}$, where S is the number of matching sequences and L is the sum of length of the two strings.

Algorithm 3 Pseudo-code for the string similarity function

```
Input: word1, word2
Output: String Similarity score between word1 and word2
$totalLength \leftarrow len(word1) + len(word2)$
$matches \leftarrow SequenceMatcher(word1, word2)$ {//a Python function returns list of matching sequences}
$simScroe \leftarrow 2 * matches / totalLength$
return $simScroe$
```

2.8.6 Sentence Similarity Function

The function `SentenceSimilarity()`, in the *Response* class, measures the similarity between two sentences. It returns a score between $[0, 1]$. Recall that a sentence is made up of *Word* objects. The function takes two sentences, the value of α (which decides the relative contributions of semantic and word order similarity to the overall similarity computation), and word similarity threshold value as inputs. Then it computes the similarity of the two sentences using semantic word similarities and word order similarities. It firsts constructs an $m \times n$ semantic similarity matrix, where m is the number of words in sentence1, and n is the number of words in sentence2, after they have been preprocessed. The i, j entries in the matrix are the semantic similarity score computed between each word in sentence1 and sentence2. After that, `aggregate_similarity()` function aggregates the similarities for each sentence with respect to the other, as described in section 2.4. `aggregate_similarity()` then returns the semantic similarity score and word order similarity score. The semantic similarity scores and the order word similarity scores are then averaged. The final score returned by `SentenceSimilarity()` is defined as a combination of the semantic similarity and word order similarity. Algorithm 4 and 5 illustrates this function.

Algorithm 4 Pseudo-code for the sentence similarity function

```
**Input**: sentence1, sentence2, $\alpha$, word similarity threshold
**Output**: Overall sentence Similarity score between sentence1 and sentence2
$m \leftarrow len(sentence1)$
$n \leftarrow len(sentence2)$
$similarityMatrix \leftarrow BuildMatrix(sentence1, sentence2, threshold)$
$semSim1, orderSim1 \leftarrow aggregate\_similarity(similarityMatrix, sentence1)$
$semSim2, orderSim2 \leftarrow aggregate\_similarity(T(similarityMatrix), sentence2)$
{
the matrix transposed}
$semSim \leftarrow (semSim1 + semSim2)/2$
$orderSim \leftarrow (orderSim1 + orderSim2)/2$
$\beta \leftarrow 1 - \alpha$
$overallSim \leftarrow \alpha * semSim + \beta * orderSim$
**return** $overallSim$
```

Algorithm 5 Pseudo-code for the aggregate_similarity function

Input: The similarity matrix (similarityMatrix) and the sentence T.
Output: Semantic similarity score and word order similarity score

```
$simSum \leftarrow 0.0$
$icsum \leftarrow 0.0$
$r_1 = []$ {//word order vector1}
$r_2 = []$ {//word order vector2}
for $row$ in similarityMatrix do
    $max \leftarrow getMaxIndex(Matrix, row)$ {//returns the column index with maximum similarity value in row $row$}
    $sim \leftarrow similarityMatrix[row][max]$
    if $simScore > 0$ then
        $ic \leftarrow T[row].ic()$ {//information content for the word in row $row$}
        $simSum+ = sim * ic$
        $icSum+ = ic$
        $r1.append(row)$
        $r2.append(max)$
    end if
end for
$simScore \leftarrow simSum/icSum$
$orderScore \leftarrow cosSim(r1, r2)$
return $simScore, orderScore$
```

Chapter 3

Experiment & Results

In order to evaluate the performance of the system, I conducted several experiments designed to answer the following research questions:

1. How does the system perform in comparison to human graders?
2. What features of an answer made it easy or difficult to grade automatically?

3.1 Datasets

To answer these questions, I used two datasets. The first dataset contained four short answer questions from an introductory statistics course, with answers provided by undergraduate students taking the course at Macalester College. The second dataset I obtained from a similar study[1], contained questions from an introductory course in computer science with answers from an undergraduate class as well. In both datasets, each question had at least 30 possible answers.

Questions in the datasets had slight variations in the degree of openness (i.e required different answer lengths). However, all of them fell under the

[1]This dataset can be downloaded from http://lit.csci.unt.edu/index.php/Download

short answer question category. Some of them required short phrase responses (2-5 words in length), and others required the generation of short explanatory responses (1-2 sentences in length).

Table 3.1 shows a sample of the dataset and includes the grades that were assigned by the two instructors.

Question: Why would you want to include an interaction term in a model?		
Model Answer: When one explanatory variable is modulated by the value of another explanatory variable.	4	4
An interaction term is necessary when one explanatory variable influences how another explanatory variable affects the model.	4	4
when one model term modulates how another model term affects the response variable.	4	4
Because we want see whether there is certain relations between our explanatory variables	1	0
Question: What does a class definition includes?		
Model Answer: Data members (attributes) and member functions	5	5
Data members and member functions	5	5
Data and functions	5	5
A class definition typically includes function definitions.	3	2

Table 3.1: *Two sample questions from the dataset used in these experiments, with the grades assigned by the two instructors*

3.2 Inter-Rater Agreement

To address the first question: *How does the system perform in comparison with human graders?* each candidate answer in the dataset was independently graded by two instructors, and automatically graded by the system using the corresponding model answer provided by the course instructor. Then, in order to evaluate the results, I measured the inter-rater agreement, which reflects the degree of agreement among graders. Several measurements that can determine the inter-rater agreement. The most common measure is the Pearson Correlation Coefficient (r), which reflects the degree of linear relationship between two variables. It ranges from +1 to −1. A correlation of +1 (and −1) means that there is a perfect positive (and negative) linear relationship between variables. A correlation of 0 means no correlation at all.

The correlation was computed between the two instructors; between each instructor and the system; and between the average of the two instructors and the system on a per question basis. Every candidate answer for a particular question is considered an independent data point in the correlation, which places more emphasis on the accuracy of the grade given to each answer.

For example, let

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

be the grades assigned by the first instructor to all answers in a question, so that each entry represents a grade for an answer. Let

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

be the grades assigned by the second instructor for the same question. Then the Pearson Correlation Coefficient is computed in the following way:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{Y})^2}} \tag{3.1}$$

Where $\bar{X}$ and $\bar{Y}$ are the mean of X and Y respectively. Note that $(X_i - \bar{X})(Y_i - \bar{Y})$ is positive if and only if X_i and Y_i lie on the same side of their respective means. Thus the correlation coefficient is positive if X_i and Y_i tend to be simultaneously greater than, or simultaneously less than, their respective means.

A weakness in the Pearson correlation to keep in mind is that it is sensitive to the data distribution. That is, it always assumes normally distributed values. However, in the datasets at hands, sometimes all candidate answers scored a narrow range, e.g all answers were pretty good. At one extreme, where all answers have the same score, the correlation would be close to zero even if the two sets of score being compared were identical. This would not correctly reflect the actual level of agreement.

3.3 Results and Discussions

Table 3.2 summarizes the correlation results for each of the categories explained above. The correlation between the two instructors was measured on average at **0.65**. This correlation can be thought of as an upper-bound for the performance expected from the system on this dataset. The system achieves a Pearson correlation of **0.51** with the average instructor grading. Considering that the upper bound was measured at **0.65**, it reasonable to say that it is performing well at **0.51**. The value of correlation between the system and human grader ranged from **0.26** to **0.8**. A closer examination of these results explains why some correlations were high and why some were too low.

Problem ID	H1-H2	H1-S	H2-S	Avg(H1&H2)-S
Q8	0.95	0.74	0.74	0.74
Q1	0.91	0.68	0.74	0.74
Q14	0.88	0.64	0.49	0.57
Q16	0.87	0.5	0.4	0.41
Q17	0.86	0.44	0.51	0.45
Q2	0.73	0.71	0.71	0.71
Q10	0.73	0.66	0.49	0.62
Q11	0.72	0.22	0.4	0.33
Q12	0.67	0.59	0.46	0.54
Q4	0.66	0.5	0.28	0.34
Q7	0.58	0.57	0.44	0.45
Q13	0.54	0.38	0.12	0.3
Q9	0.47	0.56	0.82	0.8
Q5	0.46	0.29	0.27	0.26
Q3	0.41	0.23	0.29	0.32
Q6	0.36	0.61	0.72	0.68
Q15	0.34	0.49	0.87	0.56
Average	**0.65**	**0.52**	**0.52**	**0.51**

Table 3.2: *Summary of the correlation results. H1-H2 refers to inter-human correlation, H1-S: human1-system correlation, H2-S: human2-system correlation, Avg(H1&H2)-S: average human-system correlation*

Figure 3.1 shows a scatter plot of the average of human grades versus the system grades assigned for question 1 from the dataset. The plot also shows a straight line. This is called the *best-fit* line because it comes as close to all the points on the plot as possible. If both graders agreed on all the grades, this line would be diagonal and would touch every point in the plot, giving a perfect correlation score of 1.

One observation that can be made is that there are only 4 visible data clusters. Initially, the points in each of the clusters were stacked above each other, indicating high level of agreement. So I added a little bit of jitter to each point to make them more visible (as can be seen in the graph). Also notice that the 4 visible points are spread out on the grading scale resulting in a fairly diagonal line, which is why the resulting correlation value is high (r = **0.80**).

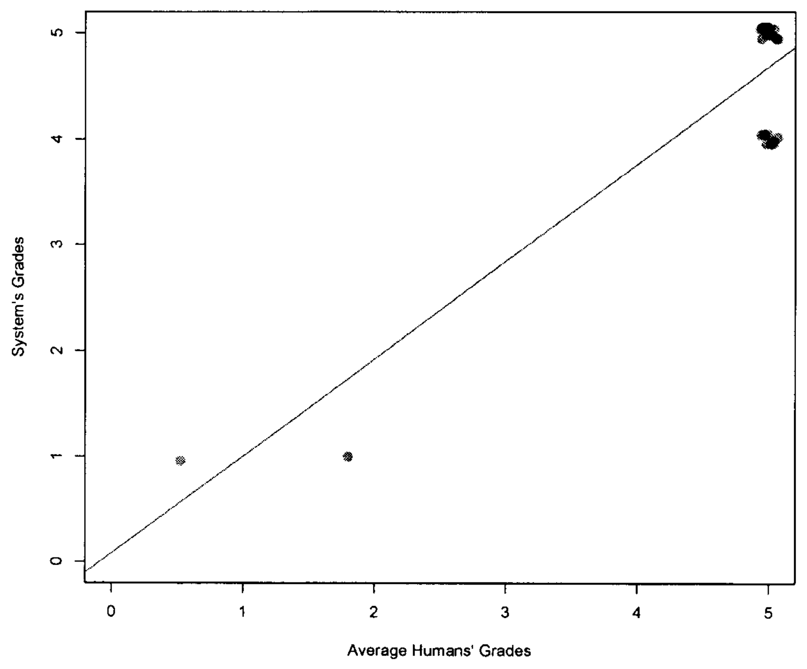

Figure 3.1: *Correlation between Average Human and the system for a question*

Figure 3.2 shows another plot similar to figure 3.1 but for question 3. Notice how all grades are heavily skewed toward the high end of the scale, and no grades at all are in the low end of the scale. Although the differences between the system grades and average human grades are less than one, the correlation score is very low (**r** = **0.32**). This is because the grades are not well spread along the scale (i.e all of grades are within a very narrow range).

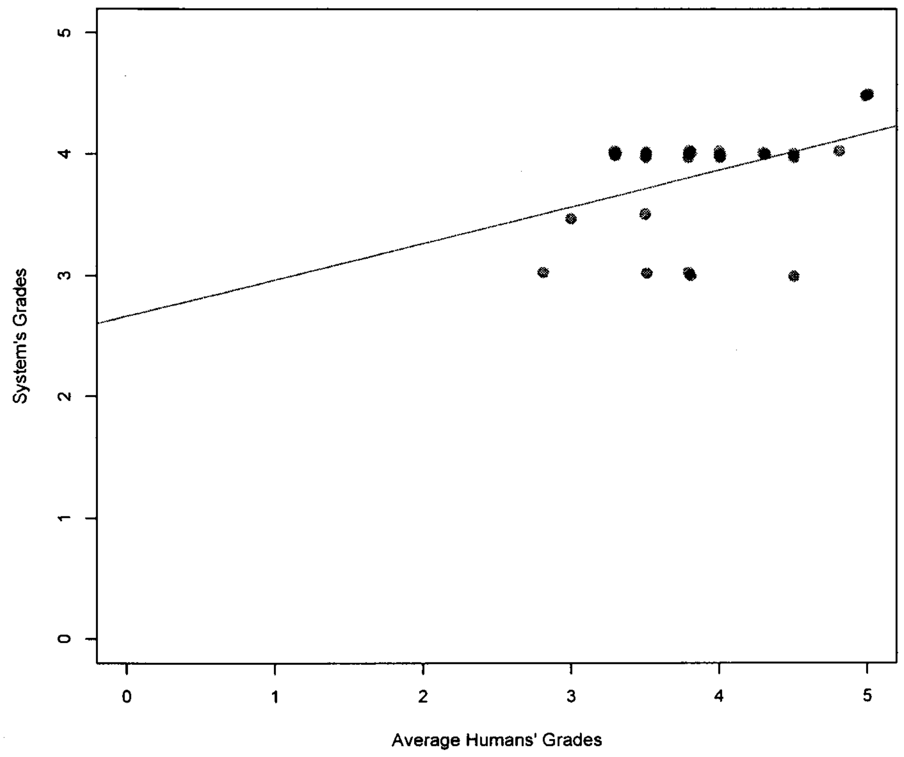

Figure 3.2: *Correlation Between Average Human and the system for a question*

Figure 3.3 shows the scatter plot in figure 3.2 but with a new data point added to the lower end of the scale. Notice after the new point is added, the best-fit line is more diagonal, indicating a better correlation (r = **0.76**). This emphasizes the point that the Pearson Correlation is very sensitive to the distribution of the data. In order to reflect the good results, the data has to be spread out.

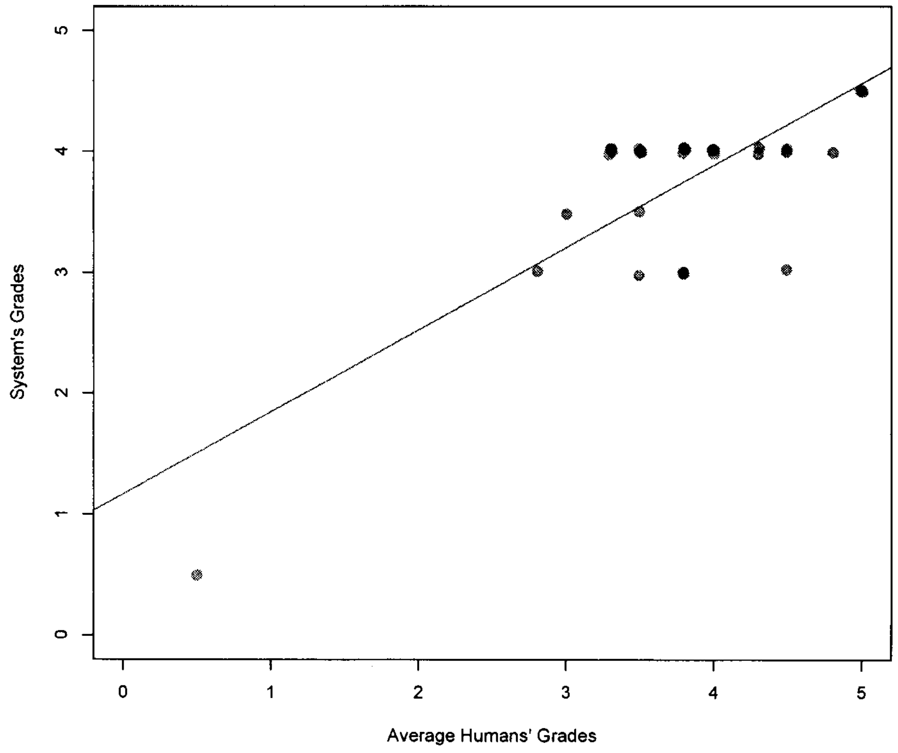

Figure 3.3: *Correlation Between Average Human and the system for a question*

3.3.1 Distribution of the Difference in Grades

Because of the weakness in the Pearson Correlation described above, I used an additional measure to evaluate my data. I computed the difference between the average human grades and the system grades for every individual candidate answer for all questions. Then I plotted the results in histogram to show the distributions of the differences.

Figure 3.4 shows a histogram of the distribution of the differences between the average of the two humans' and system's grades for all the candidate answers from all the questions. On the x-axis is the difference between the average of the two humans and system grade. Along the y-axis is the density of data points within that bin.

Notice that an envelop of the histogram is Gaussian distribution with a mean of approximately 1 and with inter-quantile range from **0.0** to **1.0**. That is, the differences seem to follow normal distribution, which implies that the errors are random and do not follow systematic patterns. The histogram also shows some outlier cases, in which difference is larger than 1.0. I will examine some of these cases in details in the next section.

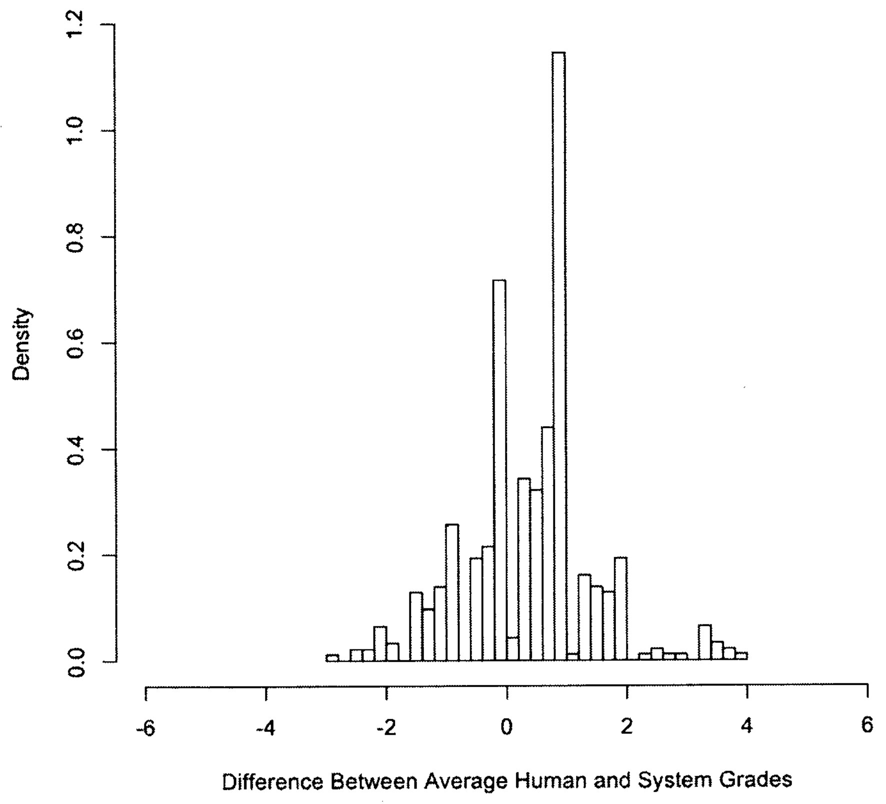

Figure 3.4: *Histogram of the distribution of the differences between the average human grades and system grades for all candidate answers*

3.4 System Weaknesses

There are times when the system did not reach a high level of correlation with human graders. This is evident by figure 3.4, which shows that there are cases with large difference between human and system grades. From the system's standpoint, there are several recognizable sources of errors, which may account for these large differences. Below I discuss these errors, addressing the third research question: *"What features of the answer undermine and support the ability to grade automatically and accurately?"*

3.4.1 Unique Answers

Two source of failures are when the answer is truly original and completely different from the model answer, and sometimes it contains idioms that make it hard for the system to infer the actual meaning. For example, one question from the dataset asks *"What is the scope of static variables?"*. The model answer provided is *"File scope"*. One student responded with *"The whole code"* and therefore received a grade of 0 from the system, whereas both instructors awarded it a grade of 4.5.

Certainly, the model answer may be edited to include that unique phrase, but the chances of it ever being encountered again are slim. More importantly, the more open-ended a question is, the more difficult it is to build a model. When the concept sought by the instructor is imprecise, then there are sure to be ways of stating it that are not similar at all to the model answers.

On the other hand, one could look at this problem as if the system is insisting on greater accuracy, offering less leniency than humans, which may not necessarily be a bad thing. Instead of trying to make the system clever enough to accept divergent responses, we can then constrain the student to give more accurate answers.

3.4.2 Misspelled Words

The system cannot recognize misspelled words, while a human can easily recognize them and deduce what the student is trying to say. For example, a question has the word *function* in its model answer. Both instructors accepted the word *funtions* as a variant of *function* and awarded the answer full score, whereas my system did not recognize the misspelled word, and hence awarded the answer partial score.

This problem may be overcome by incorporating a spell checker and corrector into the system, which would detect misspelled words and try to replace them with reasonably close words. However, this may not fully solve the problem. Some misspellings happen to be perfectly good English words. For example, in looking for the word *Data* in a response, both instructors accepted *Date* as a variation. But since *Date* is a correctly spelled English word, a spell checker will not attempt to "correct" it to match a word in the model answer. So the system is not able to recognize the answer as correct. One way to avoid this problem is measure edit distance for words that have very low semantic similarity score in the candidate answer.

3.4.3 Contradictory Information

Some answers included a correct statement supplemented by an incorrect statement that disqualifies the answer. In such cases the student is responding with a 'list' of possible answers, implying that he/she does not know the actual answer and trying to "hit the target". Incorrect statements that negate a correct statement should result in a reduction of the grades awarded.

For example, one part of a question asked about the *"scope of local variables"* in programming languages. The model answer was *"Local variables can only be accessed in its member function"*, and one student responded with *"a local variable is declared inside a function and has to be static"*. Although the first part of the answer is correct, the second part, that *"a local variable has to be static"*, negates the first statement. Both instructors took into account the additional incorrect element and gave the answer a lower grade. On the other hand, the system failed to detect such contradiction, and as a result awarded the answer a higher grade; because its wording is very similar to the first part of the model answer.

Approaches to this problem require a more sophisticated textual analysis to be able to make inferences and detect contradictions.

3.4.4 The use of Negations

Some answers contained negations, such as *not*. In such cases, the wording of the candidate answer is very similar to the model answer, but it has opposite meaning to the model answer, for example by adding (*not*). For example, one question asked about the *"declaration of global variables"*. One student responded with *"global variables may not be declared inside the class"*. Neither instructors accepted this answer, whereas the system ignored the negation and award the answer a higher grade. To avoid this problem, the system needs to be able to recognize negations and make inferences.

To summarize, the performance of the system degraded when it encountered answers with:

- Truly unique phrases and idioms.
- Misspelled words.
- Contradictory information
- Negations

With further development, most of these problems can be overcome. I will discuss possible solution in section 4.2. However, the problem of contradictory information is probably the most challenging one. It requires the computer be able to reason and make inferences, which in turn require a more in-depth analysis of the answer text.

Overall, the system performed quite well when the model answer was very concise and did not repeat words from the question statement, as well as when the question require the shorter response (1-2 phrases).

Chapter 4

Conclusions & Future Work

4.1 Conclusions

In this work, I developed an automated system for grading free-text responses for short answer questions. The system automatically assigns a grade to a *candidate answer* based on comparisons with a *model answer*. The comparison is based on the similarity of the two texts at the semantic level. It determines the semantic similarity between the two texts by examining the similarities in isolated word meaning and word order between a short answer and a model answer.

Semantic word similarity is derived from the WordNet lexical database, which models how humans organize their lexical memories. A corpus reflects the actual usage of language and words. Thus, the semantic similarity not only captures common human knowledge, but it is also able to adapt to an application area using a corpus specific to that application. Word order similarity is derived by measuring the correlation between the two word order vectors, formed based on the relative position of each word in each text. The overall similarity is a combination of semantic similarity and word order similarity.

To evaluate the performance of the system, I computed the correlation between human-assigned grades and the system assigned grades using the Pearson correlation coefficient. The correlation between the two instructors was measured on average at **0.65**. Considering the correlation between human graders as upper bound, the system achieves a good Pearson correlation of **0.51** with the average human grader.

At the same time, my system is not perfect. It failed to capture the similarity in certain cases, such as answers that contained negations, truly unique phrases or idioms, misspelled words, or contradictory information.

Automatic grading offers the potential of speed and consistency of grading decisions; human grading adds professional judgment and inference which gives credit to badly expressed understanding. The question is which one is more "objective"? While most of the answers were graded correctly by both human and the system, it is the ambiguous borderline responses that pose problems to both.

In conclusion, automatic grading of short answers using semantic text similarity is a promising approach and deserves further development.

4.2 Future Work

The overall semantic text similarity approach can, with further development, lead to accurate grading of short answer responses across a range of item types and complexities. Below I discuss directions for future work.

4.2.1 Incorporating Spell Checker

Future development can include incorporating a spell checker, so that it can recognize misspelled words.

4.2.2 More In-Depth Analysis

I can expand the analysis of the text to include more sophisticated representations of sentence structure, such as semantic parse trees, which should may allow for more effective measures of text similarity. I can also incorporate basic logical features in the text (i.e., AND, OR, NOT) to help detect negation and potential contradictory information

4.2.3 Investigating Semantic Word Similarity Metrics

There is extensive literature on semantic word similarity. In this project I used Li et al. (2003). I can compare the performance of the system with other word similarity metrics and see which one might do better job.

4.2.4 Investigating Other Correlation Metrics

I used the Pearson Correlation Coefficient to measure correlation with human graders. I can expand this analysis to include other correlation metrics such, Kappa Coefficient or Spearman's Rank Correlation Coefficient to see if they better reflect the performance of the system.

4.2.5 Investigating Other Word Order Similarity Metrics

I used the cosine similarity to determine the similarity between the two word order vectors. I can investigate other metrics such as the longest monotonically increasing sequence, and the "edit distance" algorithm at the sentence level, to consider how many changes it would take to re-tag or re-order a sentence to make it similar to another sentence.

4.2.6 Expanding the Dataset

In this study, the datasets used to evaluate the system were only from the Statistics and Computer Science fields. I can include questions from other fields such as Biology and Physics, and compare the performance of the system across multiple domains

4.2.7 Implementing a Parallel System

Many tasks in this system, such as determining sentence similarity, can be concurrently simultaneously. I can incorporate multi-threads so that these independent tasks can run in parallel.

Bibliography

Bird, S., E. Klein, and E. Loper (2009, June). *Natural Language Processing with Python* (1 ed.). O'Reilly Media, Inc.

Bloom, B. S. (1969, June). *Taxonomy of Educational Objectives: The Classification of Educational Goals.* Longman Group United Kingdom.

Bollegala, D., Y. Matsuo, and M. Ishizuka (2007). Measuring semantic similarity between words using web search engines. In *Proceedings of the 16th international conference on World Wide Web,* Banff, Alberta, Canada, pp. 757–766. ACM.

Budanitsky, A. and G. Hirst (2001). Semantic distance in WordNet: an experimental, application-oriented evaluation of five measures. In *Proceedings of the NAACL 2001 Workshop on WordNet and other lexical resources,* Volume 2.

Burstein, J., C. Leacock, and R. Swartz (2001). Automated evaluation of essays and short answers. *Loughborough University, UK, Learning & Teaching Development, Loughborough University,* 41—45.

Callear, D., J. Jerrams-Smith, V. Soh, D. J. Jerrams-smith, and H. P. Ae (2001). CAA of short Non-MCQ answers. In *Proceedings of the 5th International CAA conference.*

Foltz, P. W., D. Laham, and T. K. Landauer (1999). Automated essay scoring: Applications to educational technology. In *Proceedings of World Conference on Educational Multimedia,* Chesapeake, VA, pp. 939–944.

Francis, W. N. and H. Kucera (1979). Brown corpus ManualRevised and amplified.

Hatzivassiloglou, V., J. L. Klavans, and E. Eskin (1999). Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. In *Proceedings Of The 1999 Joint SIGDAT Conference On Empirical Methods In Natural Language Processing And Very Large Corpora*, pp. 203—212.

Islam, A. and D. Inkpen (2008). Semantic text similarity using corpus-based word similarity and string similarity. *ACM Trans. Knowl. Discov. Data* 2(2), 1–25.

Islam, M. A. and D. Inkpen (2006). Second order co-occurrence PMI for determining the semantic similarity of words. In *Proceedings of the International Conference on Language Resources and Evaluation, Genoa, Italy*, Genoe, Italy, pp. 10331038.

Jiang, J. J. and D. W. Conrath (1997, September). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of ROCLING X*, Taiwan. In the Proceedings of ROCLING X, Taiwan, 1997.

Jurafsky, D. and J. H. Martin (2008, May). *Speech and Language Processing* (2 ed.). Prentice Hall.

Kanejiya, D., A. Kumar, and S. Prasad (2003). Automatic evaluation of students' answers using syntactically enhanced LSA. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing - Volume 2*, pp. 53–60. Association for Computational Linguistics.

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

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

Leacock, C., M. Chodorow, and C. Fellbaum (1998, May). Combining local context and WordNet similarity for word sense identification. In *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*, pp. 283, 265. The MIT Press.

Li, Y., Z. Bandar, and D. Mclean (2003). An approach for measuring semantic similarity between words using multiple information sources. *Knowledge and Data Engineering, IEEE Transactions on 15*(4), 882, 871.

Lin, D. (1998). An Information-Theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, pp. 296—304.

Miller, G. A. (1995). WordNet: a lexical database for english. *Commun. ACM 38*(11), 39–41.

Miller, G. A., R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller (1990). Introduction to WordNet: an on-line lexical database. *Int J Lexicography 3*(4), 235–244.

Mitchell, T., T. Russell, P. Broomhead, and N. Aldridge (2002). Towards robust computerised marking of free-text responses. In *Proceedings of the 6th International Computer Assisted Assessment Conference*, pp. 233249.

Pulman, S. G. and J. Z. Sukkarieh (2005). Automatic short answer marking. In *Proceedings of the second workshop on Building Educational Applications Using NLP*, Ann Arbor, Michigan, pp. 9–16. Association for Computational Linguistics.

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pp. 448—453.

Sukkarieh, J., S. Pulman, and N. Raikes (2003). Auto-marking: using computational linguistics to score short, free text responses. In *Proceedings of 29th International Association for Educational Assessment (IAEA) Annual Conference.*, Manchester, UK.

Sukkarieh, J., S. Pulman, and N. Raikes (2004). Auto-Marking 2: An update on the UCLES-Oxford university research into using computational linguistics to score short. free text responses. In *Proceedings of 29th International Association for Educational Assessment (IAEA) Annual Conference.*, Philadephia, PA. International Association of Educational Assessment.

Turney, P. D. (2001). Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of the 12th European Conference on Machine Learning*, pp. 491–502. Springer-Verlag.

Wu, Z. and M. Palmer (1994). Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, Las Cruces, New Mexico, pp. 133–138. Association for Computational Linguistics.