SEMANTIC-BASED AUTOMATIC SHORT ESSAY EVALUATION AND RECOMMENDATION SYSTEM

THESES OF THE PH.D. DISSERTATION

TSEGAYE MISIKIR TASHU

SUPERVISOR: TOMÁŠ HORVÁTH, PH.D.

THIS DISSERTATION IS SUBMITTED FOR THE FULFILMENT OF DOCTOR OF PHILOSOPHY IN DATA SCIENCE AND ENGINEERING

PH.D. SCHOOL OF COMPUTER SCIENCE

HEAD OF SCHOOL: PROF. ERZSÉBET CSUHAJ-VARJÚ

PH.D. PROGRAM OF INFORMATION SYSTEMS

HEAD OF PROGRAM: PROF. ANDRÁS ID. BENČZÚR

SEPTEMBER 2020, BUDAPEST
I would like to dedicate this thesis to my loving parents . . .
Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 24,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 30 figures.

September 2020
Acknowledgements

I owe my deepest gratitude to my supervisor, Professor Tomáš Horváth who supported me throughout my studies with his wisdom, patience, encouragement, and knowledge. I really enjoyed working in a research environment that stimulates original thinking and initiative. Professor Tomáš, the skillful guidance and friendly supervision are greatly appreciated. The discussions with him were invaluable to me in sharpening my scientific skills.

This journey would not have been possible without the support of my family. I thank you for supporting me during my studies and my life in general. I am especially grateful to my son Zemariam Tsegaye for your beautiful and sweet words, which make me strong and happy when I feel lost. These acknowledgments would not be complete if I did not mention my friends and colleagues in the T-Labs ELTE (Data Science and Engineering Department, Faculty of Informatics). My laboratory was not only a source of good advice but also a source of friendship. It has been a great pleasure working with you and I appreciate your help and support.

Thanks to all my friends for sharing my joy about the beginning of my doctoral thesis and for following it with encouragement and entertainment when it seemed too difficult to finish it.
Abstract

Automated short essay evaluation (ASEE) is now widely becoming a practical solution for replacing time-consuming manual grading of student essays. Most of the traditional automated short essay scoring systems rely on supervised machine learning approaches that require carefully designed features and a large amount of manually annotated training set to evaluate and score essays (i.e. to provide summative feedback). There are cases in which it is challenging to find labeled training short answers that require a great deal of effort when creating labeled sets. It becomes difficult in situations where new responses are generated more frequently because it is challenging to decide which and how many of the newly generated short answers (from the students) need to be manually labeled by the teacher to form a reliable rating machine. To address the issues in providing summative feedback which is proving the score of the essay, we introduced two approaches. The first approach is unsupervised and uses semantic features to predict the score by only comparing the student answer with a reference answer (correct answer) called the Pairwise method. Instead of using input representation based on bag-of-words, Pairwise considers both the student answer and reference answer as a sequence of words with rich contextual structure, and it retains maximum contextual information in its latent semantic representation by projecting each word in its context onto a low-dimensional continuous feature vector using skip-gram models. Then, the Pairwise method computes the semantic similarity between the feature vector representation of the reference solution and a student answer using Relaxed word mover’s similarity (RWMS) method. The second approach is to predict the score of the essay using a small annotated short essay answers. We proposed and introduced a new and efficient way of selecting a small-sized training set to be annotated by human rater that will be used to train the scoring engine using semantic intelligent k-means
and semantic Locality Sensitive Hashing (Sem-LSH) techniques. Here, we investigate the
question of what proportion of the short essays should the teacher score manually before
the ASEE engine is trained and used on new short essays without labels (scores) and how
to select those responses proportionally.

Also, most of the current short answer scoring systems only provide a score that is
good but does not provide more information for the students in the source of the error or
about ways to improve their answer. To address the issues in proving formative feedback,
we proposed a new semi-supervised and semantic-based feedback recommendation that
provides formative feedback to short essay answers. Our proposed approach allows the
teachers to interact with the scoring engine to give feedback for short essay answers and
the system provide a recommendation to other similar essays based on their similarity to
the solution which has been evaluated by the assessor. According to our knowledge, this
work is the first and a pioneer in providing formative feedback to short answers.
Contents

List of Figures xi

List of Tables xvii

1 Introduction 1

1.1 Motivation ......................................................... 1
1.2 Research Tasks .................................................. 4
1.3 Scientific Engineering Contribution .......................... 5
1.4 Publications ...................................................... 5
1.5 Dissertation Overview and Structure ....................... 7

2 Feature Representation and Models 9

2.1 Feature Representation ......................................... 9
  2.1.1 Word Vectors ............................................... 10
  2.1.2 Bag of Words based methods .............................. 10
  2.1.3 Term Weighting Methods .................................. 11
  2.1.4 Singular Value Decomposition Based Methods .......... 13
  2.1.5 Iteration Based Methods – word2vec .................... 16
2.2 Distance or Similarity Measures ............................... 21
  2.2.1 Cosine Similarity .......................................... 21
  2.2.2 Euclidean Distance ....................................... 22
  2.2.3 Manhattan Distance ..................................... 22
  2.2.4 Jaccard similarity ......................................... 22
<table>
<thead>
<tr>
<th>2.3 Models Utilized in the Thesis</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1 Locality Sensitive Hashing</td>
<td>22</td>
</tr>
<tr>
<td>2.3.2 Word Movers Distance</td>
<td>24</td>
</tr>
<tr>
<td>2.3.3 Latent Semantic Analysis</td>
<td>26</td>
</tr>
<tr>
<td>2.3.4 Latent Dirichlet Allocation</td>
<td>27</td>
</tr>
<tr>
<td>2.3.5 Clustering</td>
<td>29</td>
</tr>
<tr>
<td>3 Pair-wise : Automatic Short Essay Evaluation</td>
<td>35</td>
</tr>
<tr>
<td>3.1 Motivation</td>
<td>35</td>
</tr>
<tr>
<td>3.2 Related Work</td>
<td>37</td>
</tr>
<tr>
<td>3.3 The Pair-Wise ASEE Approach</td>
<td>43</td>
</tr>
<tr>
<td>3.3.1 A Layered based Pair-Wise Architecture</td>
<td>45</td>
</tr>
<tr>
<td>3.3.2 Plagiarism detection using shingles</td>
<td>47</td>
</tr>
<tr>
<td>3.3.3 Foolish detection</td>
<td>47</td>
</tr>
<tr>
<td>3.4 Experiment</td>
<td>48</td>
</tr>
<tr>
<td>3.4.1 Quantitative Evaluation</td>
<td>49</td>
</tr>
<tr>
<td>3.4.2 Qualitative Evaluation</td>
<td>51</td>
</tr>
<tr>
<td>3.4.3 Comparison with the Other Systems</td>
<td>53</td>
</tr>
<tr>
<td>3.4.4 Plagiarism Detection Evaluation</td>
<td>54</td>
</tr>
<tr>
<td>3.5 Summary</td>
<td>56</td>
</tr>
<tr>
<td>4 Reducing Annotation Effort</td>
<td>57</td>
</tr>
<tr>
<td>4.1 Motivation</td>
<td>57</td>
</tr>
<tr>
<td>4.2 Related Work</td>
<td>59</td>
</tr>
<tr>
<td>4.3 Proposed model</td>
<td>61</td>
</tr>
<tr>
<td>4.3.1 Problem statement</td>
<td>61</td>
</tr>
<tr>
<td>4.3.2 Straightforward approach</td>
<td>61</td>
</tr>
<tr>
<td>4.3.3 Efficient approach</td>
<td>62</td>
</tr>
<tr>
<td>4.3.4 Semantic LSH (Sem-LSH)</td>
<td>63</td>
</tr>
<tr>
<td>4.4 Experimental Setup</td>
<td>64</td>
</tr>
</tbody>
</table>
# CONTENTS

4.4.1 Feature representation ................................................. 64  
4.4.2 Data Preprocessing ..................................................... 66  
4.4.3 Clustering ................................................................. 66  
4.4.4 Dataset ................................................................. 66  
4.4.5 Evaluation Metrics ..................................................... 67  
4.5 Results and discussion .................................................. 68  
4.6 Summary ................................................................. 70  

5 Semantic-Based Feedback Recommendation ......................... 73  
5.1 Motivation ................................................................. 73  
5.2 Related Work ............................................................. 75  
5.3 Feedback Tag Recommendation ....................................... 78  
5.3.1 Lexical Similarity ......................................................... 81  
5.3.2 Relaxed Word Movers based similarity ............................ 81  
5.4 Experimental study ...................................................... 82  
5.4.1 Dataset selection and preparation ................................ 82  
5.4.2 Data Preprocessing .................................................... 83  
5.4.3 Evaluation Metrics .................................................... 84  
5.4.4 Experimental results and Discussions ........................... 85  
5.5 Summary ................................................................. 88  

6 Conclusion and Recommendation ........................................ 89  
6.1 Contributions ............................................................. 89  
6.2 Future Research Directions ............................................ 90  

Bibliography ................................................................. 93  

Appendix A The E-test Application .................................... 105  
A.1 Motivation ................................................................. 105  
A.2 Related Works ........................................................... 106  
A.3 System Details ............................................................ 107
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.3.1 System Modules</td>
<td>107</td>
</tr>
<tr>
<td>A.3.2 System Work Process</td>
<td>110</td>
</tr>
<tr>
<td>A.3.3 System Implementation</td>
<td>110</td>
</tr>
<tr>
<td>A.4 Automatic scoring Process</td>
<td>110</td>
</tr>
<tr>
<td>A.5 System Evaluation</td>
<td>111</td>
</tr>
<tr>
<td>A.6 Summary</td>
<td>112</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Graphical model representation of LDA. The boxes are “plates” representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document [12]. 29

3.1 The architecture of the proposed Pair-Wise ASEE approach. 46

3.2 A quantitative comparison using \( nRMSE \) (Equation 3.5) of the proposed Pair-Wise approach and the baselines using LSA, WordNet and Cosine similarity. In case of the datasets no. 3, 4, 6, 7 and 8, the average performance from 10 runs (corresponding to the 10 randomly chosen reference solutions) is reported while in case of the other five datasets (where only the one reference solution indicated in the data is used), the result from one run is reported. 50

3.3 A qualitative comparison using \( prank \) (Equation 3.6) of the proposed Pair-Wise approach and the baselines using LSA, WordNet and cosine similarity. In case of the datasets no. 3, 4, 6, 7 and 8, the average performance from 10 runs (corresponding to the 10 randomly chosen reference solutions) is reported while in case of the other five datasets (where only the one reference solution indicated in the data is used), the result from one run is reported. 52

4.1 The classical supervised scoring approach where a subset of instances are selected for manual annotation and then training the model. 59
LIST OF FIGURES

4.2 The classical supervised scoring approach with a clustering where a subset of instances from each cluster is selected proportionally for manual annotation then training the model. .................................................. 59
4.3 Graphical model representation of Sem-LSH ........................................ 64
5.1 Normalized Confusion matrix which shows the number of the cases from actual recommendation are correctly predicted (recommended) and how may of the cases are incorrectly predicted (recommended) by the RWMS algorithm .................................................. 86
5.2 Normalized Confusion matrix which the number of the cases from actual recommendation are correctly predicted(recommended) and how may of the cases are incorrectly predicted (recommended) by the Lexical based algorithm .................................................. 86
5.3 Normalized Confusion matrix which show the number of the cases from actual recommendation are correctly predicted(recommended) and how may of the cases are incorrectly predicted (recommended) by the Lexical based algorithm .................................................. 87
A.1 The System Management Module (Maintenance) of our system. ........ 108
A.2 List of modules and sub-modules available on our system for teachers use. 108
A.3 List of modules and sub-modules available on our system for students use. 108
A.4 Our system’s Online Exam Management module: an illustration to show how the teachers can create essay exams. ................................. 109
A.5 Our system’s Online Exam Management module: an illustration to show how the students can submit their answers. ................................. 109
A.6 Our system’s feedback module: Showing how the teachers can give textual comments by selecting some part of the essay. ................................. 109
A.7 Our system’s feedback module: Shows how the students can see textual comments given by the teacher. ................................. 109
A.8 The Work Process of our online exam management and evaluation system. 110
LIST OF FIGURES

A.9 Overall, how satisfied or dissatisfied are you with our system? . . . . . . . 111
A.10 How well do our system meets your needs? . . . . . . . . . . . . . . . . 111
A.11 Which of the following words would you use to describe our system?

Select all that apply. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 111
A.12 How much time do you save while evaluating students essay with the system? 111
List of Tables

3.1 Essay sets used in the experiment and their main characteristics . . . . . 49
3.2 The p-values resulting from the Wilcoxon signed-rank test between the $nRMSE$ results of the proposed ASEE and the baselines using LSA, WordNet and Cosine similarity. ................................. 51
3.3 The p-values resulting from the Wilcoxon signed-rank test between the $prank$ results of the proposed ASEE and the baselines using LSA, WordNet and Cosine similarity . ........................................... 53
3.4 Accuracy comparison of various systems from the Kaggle competition and the literature. ................................................................. 54
3.5 Shingle based plagiarism detection results ........................................ 55
4.1 Essay sets used in the experiment and their main characteristics . . . . . 67
4.2 The resulting accuracy of the proposed LSH-based method using QWK compared with different other two baseline. .......................... 69
5.1 Performance of feedback tag based recommendation using Precision, Recall and F1-Score for dataset ....................................................... 85
A.1 Survey results collected from the three questions which have the same answer choices (Q1-Does the system help you grade more fairly?, Q2-Does the system save you time in grading? and Q3-Does the system make the evaluation of essay questions more easily?). .......................... 112
Chapter 1

Introduction

1.1 Motivation

Evaluation plays a central role in the educational process, as it determines whether the educational goals and standards of the lessons are being met. Most researchers in educational assessment agree that objective type questions, requiring specific answers without an option to provide own opinion, fail to measure learning outcomes that require the ability to retrieve, organize and integrate ideas, and the ability to express oneself in writing [55, 96]. Essays or subjective type questions serve their most useful purpose in measuring such results, which correspond to the higher levels of Bloom’s [11] taxonomy. Many researchers see essays (long or short) or subjective typology exams as the most useful tool for assessing learning outcomes, including the ability to remember, organize and integrate ideas [96]. The interest in the development and use of automatic evaluation systems has grown in recent years due to different reasons:

1. Inconsistency and inaccuracy in scores assigned by human markers due to factors such as fatigue, loss of concentration arising from boredom, appearance (e.g. neatness of handwriting) and assessors tendency to avoid extreme categories of a rating scale.

2. Variables in the markers’ background, such as the level of professional experience, prior knowledge in marking, linguistic and cultural backgrounds, perceptions of
language proficiency and assumptions of language acquisition can all influence markers’ marking behaviour and judgment.

3. The other problem associated with manual scoring is the time it takes to complete the marking process. This leads to significant delays in communicating scores and other feedback to schools, students and teachers. When results are returned to schools promptly, teachers have the option of using the results to plan better and to tailor lessons to the specific needs of students. The students can use the results to gain a sharper understanding of their weaknesses and adjust their study patterns accordingly. But when results are delayed because of the time it takes to mark the responses, the instructional value of using the test results to improve learning is reduced.

4. The possibilities provided by e-learning approaches to asynchronous and ubiquitous education.

These problems can be addressed by introducing automated assessment tools for essays. An automated evaluation system would at least be consistent in the way it evaluates essays, and enormous cost and time savings could be achieved if the system could be shown to evaluate essays in the range of those awarded by a human assessor. Since its introduction by Ellis Page and his colleagues in 1960s [71], a number of research activities have been carried out [71, 4, 114, 82] in evaluating essays for assessing writing skills however there are limited researches in the automated evaluation of short essays which are generated more frequently. The task of scoring essays is regarded as a machine learning problem that learns to approximate the assessment process using handcrafted features with supervised learning [5, 30, 25].

Short essay responses are widespread in many educational scenarios. It offers the advantage that a student must independently formulate an answer and thus show that he/she has understood the material. For almost all short essay type questions, the most natural way to give an answer would be to use natural language. However, the automatic processing of natural language responses suffers from the problem of language variation: there is typically an unlimited number of expressions of the same idea. The enumeration
of all correct solutions is often possible for minimal tasks such as filling a blank with a suitable grammatical form, but language variation makes it impossible to list all the correct solutions that are not possible for many tasks where the content of an answer is to be evaluated.

Therefore, automatic evaluation of the semantic content of a short essay response is an essential task from an educational point of view since human scoring can be a time-consuming and lengthy task. It is also a challenging task from a natural language processing (NLP) perspective. Short-essay questions have become famous for automatic evaluation because of their medium linguistic complexity: They have a complexity that goes beyond closing a gap with a single word, both in terms of response length and language variance in the response, but is limited compared to longer texts.

Most of the state-of-the-art automatic short essay evaluation (ASEE) systems rely on supervised machine learning approaches that require a large number of manually annotated training sets to train the scoring engine. There are cases in which it is challenging to find labelled training short answers that require a great deal of effort when creating labelled sets. For the evaluation of high-stakes short answers, the effort invested in manually marking a large number of short answers to train a good model may be justified [110], but it becomes difficult in situations where new responses are generated more frequently because it is challenging to decide which and how many of the newly generated short answers (from the students) need to be manually labelled by the teacher to form a reliable rating machine. With manual annotation, the instances to be scored are selected at random. There is a problem, however, in such process: Annotators will inevitably see many similar answers that will not add much new knowledge to the trained model and the class distribution in the data is often highly distorted [110]. Researches that investigate on how to select instances to be annotated manually and, also, ways on how to reduce the number of annotated training examples required to train a model must be done to make ASEE systems more useful. The investigation on ASEE should, therefore, focus on how to replace the random selection of responses to be annotated with a more informal and convenient approach and
also to investigate on the possibility of designing unsupervised semantic-based scoring systems that does not require any labeled data.

On the other hand, the purpose of the automatic assessment systems is not only to shorten the time spent by the teachers in the evaluation process and providing consistent scores, but also to overcome challenges in providing timely formative feedback to learners in the form of textual comments as noted by AL-Smadi and Gütl [1]. According to Gibbs and Simpson [31] feedback should be appropriate to the aim of the assignment and its criteria, appropriate in relation to students’ conception of learning, their knowledge and of the discourse of the discipline attended to and acted upon. Therefore, the research in automatic short answer grading systems should also focus on designing an intelligent method that allows assessors to highlight parts of the response and provide feedback where formative feedback is required.

1.2 Research Tasks

In light of the above discussion about the existing models’ weaknesses, the following research tasks have been identified:

1. Task one ($T_1$): Given a set of student essay answers ($S_a$) and their corresponding reference answers ($R_a$), the task of pairwise scoring is to learn a similarity function $sim(S_a, R_a)$ that predicts the score by computing the semantic similarity between the student answer and reference answer.

2. Task two ($T_2$): Given a set of student essay answers ($S_a$) with their corresponding text selected as an error ($E_r$) and teacher feedback ($E_f$), the task of feedback recommendation is to learn a model that learns a function that tries to give formative feedback in the form of recommendation to other new student essays written on the same topic.

3. Task three ($T_3$): Given a set of student essay answer ($S_a$) and their corresponding teacher score or label ($S_c$), the task is to design a function that helps teachers
in selecting the most representative essays for manual annotation and to learn a scoring function that scores essay semantically by training a scoring engine on small representative essay sets.

### 1.3 Scientific Engineering Contribution

Based on the above identified research tasks, the following scientific contributions are presented in this dissertation:

1. Unsupervised semantic-based short essay scoring using neural word embedding called pair-wise: The proposed Pair-wise method computes the semantic similarity between the reference answer and a student answer using a relaxed word movers similarity. A new qualitative evaluation method called prank was also introduced [88, 89].

2. A new semantic and semi-supervised approach in scoring short essays automatically: We investigate the question of what proportion of the short responses should the teacher score manually before the ASE engine is used on new short essays and how to select those responses proportionally. Thus, this study introduces a new way of addressing the issue of reducing the workload of human graders in creating training sets for semi-supervised ASE [92, 91].

3. Semi-supervised and semantic-based feedback recommendation: We introduce a semantic-based feedback recommendation approach for automatic short essay grading that will allow the assessors to interact with the system, allow them to give feedback and give the recommendation to other similar essays based on their similarity to the solution which has been evaluated by the assessor [90].

### 1.4 Publications

List of publications, in chronological order, used in the dissertation:
Introduction


   • The paper received the “Best Student Paper” Award of the conference.


   • The paper was published in a special issue of selected and extended papers from the CSEDU 2018 Conference.


   • The paper was nominated for the “Best Short Paper” Award.


Other publications of the author:


### 1.5 Dissertation Overview and Structure

The main aim of this dissertation is to build a semantic based ASEE that would not only improve the grading accuracy but would be capable of providing formative feedback. Thus, we first introduced unsupervised and semantic based short essay scoring and then introduce approaches for reducing workload in manual annotation and for providing formative feedback. In chapter two, we first describe several techniques for representing a short essay and the state-of-the-art models used in automatic short essay scoring. We present a review to the state-of-the-art which are relevant to each chapter. In Chapter three, we introduce the pairwise approach to short essay scoring. Chapter four presents our work towards reducing annotation effort in semantic automatic short essay evaluation. The fifth chapter is where we present and describe our novel work in providing formative feedback to the students’ short essay. Chapter six draws conclusions.
Chapter 2

Feature Representation and Models

2.1 Feature Representation in Short Essays

Representation is the first, probably the most important and most complex problem in almost all Natural Language Processing (NLP) tasks, and it is the way we represent words as input for any of our models. Representation aims to numerically represent unstructured text documents as word vectors to make them mathematically computable. Given a set of text documents $D = \{d_1, d_2, \ldots, d_n\}$, where each $d_i$ stands for a document, the objective of text representation is to represent each $d_i$ in $D$ as a point $v_i$ in a vector space or numerical space $V$, where the distance/similarity between each pair of points or vectors in the space $V$ is well defined. In order to be good at most NLP tasks in general and essay evaluation in particular, we must first have an idea of similarity and difference between words. With word vectors, we can easily encode this capability in the vectors themselves (with distance measurements like Jaccard, Cosine, Euclidean, and others). Therefore, how we represent inputs affects how an autonomous learning agent can analyze them and extract insights, make predictions and deliver knowledge. In the upcoming sub-sections, we will present methods of text document representation.
2.1.1 Word Vectors

**One-hot vector:** Represent every word in the document as an $\mathbb{R}^{|V| \times 1}$ vector with all 0s and one 1 at the index of that word in the sorted input document where $|V|$ is the size of the vocabulary. The term "one-hot" comes from digital circuit design, meaning "a group of bits among which the legal combinations of values are only those with a single high (1) bit and all the others low (0)" [20].

$$d = (w_1, w_2, \cdots, w_m) = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} w_2 \\ \vdots \\ w_m \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} w_m \end{pmatrix} = \begin{pmatrix} 1 \end{pmatrix} \quad (2.1)$$

It is one of the easiest way to represent the documents but the one-hot vector representation have the following drawbacks:

1. **The vocabulary size issue:** as the size of the vocabulary increase by $n$, the feature vector size also increases by $n$. One-hot vector dimensionality is the same as the number of words. More features mean more parameters to estimate, and it requires exponentially more data to estimate those parameters well enough to build a reasonably generalisable model.

2. **The computational issue:** Each word’s feature vector is mostly zeroing, and many machine learning models will not work well with very high dimensional and sparse features. With such an ample feature space, we will also be in danger of running into memory and even storage concerns, mainly if the selected models do not perform nicely with compressed versions of sparse matrices.

2.1.2 Bag of Words based methods

The bag-of-words (BOW) model is a representation used in natural language processing where a text in a document is represented as the bag of its words, disregarding grammar
and even word order but keeping multiplicity. The BOW model is used in document classification where the frequency of the word is used as a feature in building a model.

2.1.3 Term Weighting Methods

Term weighting is a procedure that takes place during the text indexing process in order to assess the value of each term to the document. It is the assignment of numerical values to terms that represent their importance in a document in order to improve retrieval effectiveness. Essentially, it considers the relative importance of individual words in text mining system, which can improve system effectiveness, since not all the terms in a given document collection are of equal importance. Weighing the terms enables the models to determine the importance of a given term in a certain document.

Term Frequency

The significance of a certain term in a given document can be represented by the term frequency of that term in the document. The simplest approach is to assign the weight to be equal to the number of occurrences of term $t$ in document $d$. This weighting scheme is referred to as term frequency and is denoted $tf_{t,d}$, with the subscripts denoting the term and the document in order. For a document $d$, the set of weights determined by the $tf_{t,d}$ may be viewed as a quantitative digest of that document. In this view of a document, known in the literature as the bag of words model, the exact ordering of the terms in a document is ignored but the number of occurrences of each term is material. We only retain information on the number of occurrences of each term. Thus, the document “Bálint is quicker than Kinga” is, in this view, identical to the document “Kinga is quicker than Bálint”. Nevertheless, it seems intuitive that two documents with similar bag of words representations are similar in content.

Inverse Document frequency

The term frequency indicates the importance of the term in a given document, but knowing the term importance in a collection of documents is also significant. Term frequency was
criticized as a method of determining term significance because in its simplest form, it treats all terms equally based on raw count, which does not take into account the term’s discriminating power. To resolve this problem, Sparck Jones [77] suggested the use of the relative collection frequency or inverse document frequency (IDF), making the frequency of the term in the collection as a whole a variable in representation. IDF places greater emphasis on the value of a term as a means of distinguishing one document from another than on its value as an indication of the content of the document itself.

Raw term frequency, as introduced above, suffers from a critical problem: All terms are considered equally important when it comes to assessing relevancy on pairwise similarity. Certain terms have little or no discriminating power in determining relevance. For instance, a collection of documents on the auto industry is likely to have the term auto in almost every document. To this end, a mechanism for attenuating the effect of terms that occur too often in the collection to be meaningful for relevance determination was introduced. An immediate idea is to scale down the term weight of term \( t \) with high collection frequency \( (c_f) \), defined to be the total number of occurrences of a term in the collection. The idea would be to reduce the \( t_f \) weight of a term by a factor that grows with its collection frequency.

Instead, it is more commonplace to use the document frequency \( d_f \), defined to be the number of documents in the collection that contain a term \( t \). This is because in trying to discriminate between documents to score it is better to use a document-level statistic (such as the number of documents containing a term) than to use a collection-wide statistic for the term. The reason to prefer \( d_f \) to \( c_f \) is that these frequencies can behave rather differently. In particular, the \( c_f \) values for both terms \( t_1 \) and \( t_2 \) are roughly equal, but their \( d_f \) values differ significantly. Intuitively, we want the few documents that contain \( t_1 \) to get a higher boost for a query on insurance than the many documents containing \( t_2 \) get from a query on try. Inverse document frequency is defined as follows:

\[
idf = \log \frac{N}{d_f} \tag{2.2}
\]
where \( N \) is the total number of documents in the collection and \( df_t \) is the total number of documents that contain term \( t \). Thus the \( idf \) of a rare term is high, whereas the \( idf \) of a frequent term is likely to be low.

**tf-idf weighting**

We now combine the definitions of term frequency and inverse document frequency, to produce a composite weight for each term in each document. The tf-idf weighting scheme assigns to term \( t \) a weight in document \( d \) given by

\[
tf-idf_{t,d} = tf_{t,d} \times idf_t
\]  

In other words \( tf-idf_{t,d} \) assigns to term \( t \) a weight in document \( d \) that is

1. highest when \( t \) occurs many times within a small number of documents (thus lending high discriminating power to those documents);
2. lower when the term occurs fewer times in a document, or occurs in many documents (thus offering a less pronounced relevance signal);
3. lowest when the term occurs in virtually all documents

**2.1.4 Singular Value Decomposition Based Methods**

Singular Value Decomposition (SVD) based methods work by first looping over a massive dataset and accumulate word co-occurrence counts in form of a (sparse) matrix \( X \) and then perform SVD to decompose \( X \) to matrices \( U \), \( S \) and \( V \) such that \( X = USV^T \). Finally, the rows of \( U \) are used as the word embeddings for all words in the vocabulary. Such approaches are known as distributional semantics in the literature, i.e. the concept of representing the meaning of a word based on the context in which it usually appears. It is dense and can better capture similarity. Let us discuss a few choices of \( X \).
Word-Document Matrix

Word-Document Matrix builds a representation with the assumption that related words will often appear in the same documents and unrelated words would probably not consistently appear together. Such facts will be used to build a word-document matrix, $X$ in the following manner: Loop over the documents and for each time word $w_i$ appears in document $d_j$, add 1 to entry $X_{ij}$. This is a very large matrix $\mathbb{R}^{|V| \times n}$ and it scales with the number of documents $n$.

Window-based Co-occurrence Matrix

The same kind of logic applies here, however, the matrix $X$ stores co-occurrences of words, thereby becoming an affinity matrix. In building window-based co-occurrence matrix, count the number of times each word appears inside a window of a particular size around the word of interest and calculate this count for all the words in the corpus.

Applying SVD to the co-occurrence matrix

Once the co-occurrence matrix $X \in \mathbb{R}^{|V| \times |V|}$ is generated, We can apply SVD on $X$, observe the singular values (the diagonal entries in the resulting matrix $S$), and cut them off at some index $k$, based on the desired percentage of variance captured, defined as

$$\frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{|V|} \sigma_i}$$

(2.4)

where $\sigma_i$ refers to the entry $S_{i,i}$. We then take the sub-matrix of $U_{1:|V|,1:k}$ to be our word embedding matrix. This would thus give us a $k$-dimensional representation of every word.
in the vocabulary. Applying SVD to $X$:

$$
|V| \begin{bmatrix} X \end{bmatrix} = |V| \begin{bmatrix} u_1 & u_2 & \cdots \end{bmatrix} |V| \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} |V| \begin{bmatrix} -v_1 & -v_2 & - \end{bmatrix}
$$

The dimensionality of the matrix $X$ can be reduced, resulting in matrix $\hat{X}$, by selecting the first $k$ singular vectors as follows:

$$
|V| \begin{bmatrix} \hat{X} \end{bmatrix} = |V| \begin{bmatrix} u_1 & u_2 & \cdots \end{bmatrix} k \begin{bmatrix} \sigma_1 & 0 & \cdots \\ 0 & \sigma_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} k \begin{bmatrix} -v_1 & -v_2 & - \end{bmatrix}
$$

Both of these methods give us word vectors that are more than sufficient to encode semantic and syntactic information but are associated with many other problems:

1. The dimensions of the matrix change very often and it is hard to incorporate new words or documents.
2. The matrix is extremely sparse since most words do not co-occur.
3. The matrix is very high dimensional, in general, and is computationally very expensive to perform SVD. The cost for computing SVD for a $m \times n$ matrix is $O(mn^2)$.

As SVD based methods do not scale well for big matrices, it is hard and computationally expensive to incorporate new words or documents. However, to mitigate the issue of high dimensional and sparse matrix, one might apply the following approaches:

1. Ignore function words such as “hey”, “he”, “has”, etc.
2. Apply a ramp window – i.e. weight the co-occurrence count based on the distance between the words in the document.

3. Use Pearson correlation and set negative counts to 0 instead of using just raw count.

The above described “hacks” can only address the problem of SVD based methods partially and therefore, methods that solve many of these issues in a far more elegant and robust manner are desirable.

### 2.1.5 Iteration Based Methods – word2vec

Rather than computing and storing global information about a huge dataset, we can create a model that can encode the probability of a word in its context. Iteration-based methods capture co-occurrence of words individually, instead of capturing all counts of the events directly like in the case of SVD based methods.

The idea is to design a model whose parameters are the word vectors. Then, train the model to a specific goal. At each iteration, we run our model, evaluate the errors and follow an update rule that punishes the model parameters that caused the error.

Several approaches have been tested. Collobert et al. [21] designed models for NLP which first step is to transform each word into a vector. For each individual task (e.g. named entity recognition, part-of-speech tagging) not only the parameters of the model were trained but also the vectors. This approach achieved excellent performance in calculating the correct word vectors. Recently, a probabilistic method, called word2vec, was introduced by Mikolov et al.[66] . The word2vec model relies on a very important hypothesis in linguistics called distributional similarity and includes the following two training methods:

- Two algorithms: continuous bag-of-words (CBOW) and skip-gram. CBOW aims to predict a center word from the surrounding context in the form of word vectors. Skip-gram does the opposite and predicts the distribution (probability) of context words from a center word.
Feature Representation and Models

- Two training methods: Negative sampling and hierarchical Softmax. Negative sampling defines an objective by sampling negative examples, while hierarchical Softmax defines an objective using an efficient tree structure to calculate probabilities for the entire vocabulary.

**Language Models (Unigrams, Bigrams)**

A good language model gives a high probability for a completely valid sentence, syntactically and semantically and, similarly, gives very low probability for sentences that make no sense. Therefore, the objective of language model is to assign a probability to a sequence of tokens. We can compute this probability on any given sequence of \( n \) words \( P(w_1, w_2, \ldots, w_n) \). We can take the unary language model approach and break apart this probability by assuming the word occurrences are completely independent.

\[
P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i)
\]  

(2.5)

However, this is a bit ludicrous because the next word is highly contingent upon the previous sequence of words, and the silly sentence example might score highly. Therefore, the unigram model can be modified by letting the probability of the sequence be dependant on the pairwise probability of a word in the sequence and the word next to it. Such a model is called a bigram model and can be defined as

\[
P(w_1, w_2, \ldots, w_n) = \prod_{i=2}^{n} P(w_i | w_{i-1})
\]  

(2.6)

Again this certainly a bit naive since it only deals with pairs of neighbouring words rather than evaluating a whole sentence, however, this representation gets pretty far along. In the word-word matrix with a context of size 1 these pairwise probabilities can be learned. Again, this would require computing and storing global information about a massive dataset. So, better models that could learn these probabilities like word2vec were proposed.
word2vec – Continuous Bag of Words Model (CBOW)

The Continuous Bag of Words Model (CBOW) is a model that predicts a center word from the surrounding context [67]. In CBOW approach, given an input sentence “The cat jump over the puddle” is to treat {“The”, “cat”, “over”, “the”, “puddle”} as a context, and, from these words to be able to predict or generate the center word “jumped”.

In CBOW, first we have to set up the known parameters such that the input sentences represented by one-hot word vectors. Let the input one-hot vector or context is represented by $x^c$ and the output as $y^c$. Since there is only one output, it can be denoted by $y$ which is the one-hot vector of the known center word. Now let us define the unknowns in the model. Create two matrices, $A \in \mathbb{R}^{n \times |V|}$ and $B \in \mathbb{R}^{|V| \times n}$ where $n$ is an arbitrary size which defines the size of the embedding space. $A$ is the input word matrix, such that, the $i$-th column $a_i \in \mathbb{R}^n$ of $A$ is the embedded vector for word $w_i$. Similarly, $B$ is the output word matrix. The $j$-th row $b_j \in \mathbb{R}^n$ of $B$ is the embedded vector for word $w_j$ when it is an output of the model and the row of $B$ is denoted as $b_j$. Note that for each word $w_i$, we want to learn two vectors (i.e. input word vector $a_i$ when the word is in the context and output word vector $b_i$ when the word is in the center).

Generally, the way how CBOW model works can be broken down as follows:

1. Generate the one-hot word vectors for the input context of size $m$, with regard to $x^c$, such that $(x^{c-m}, x^{c-1}, x^{c+1}, ..., x^{c+m}) \in \mathbb{R}^{|V|})$.
2. Get the embedded word vectors for the context $(a_{c-m} + a_{c-m+1} + ... + a_{c+m} \in \mathbb{R}^n)$
3. Average these vectors to get $\hat{a} = \frac{a_{c-m} + a_{c-m+1} + ... + a_{c+m}}{2m} \in \mathbb{R}^n$
4. Generate a score vector $Z = B \hat{a} \in \mathbb{R}^{|V|}$. As the dot product of similar vectors is higher, it will push similar words close to each other in order to achieve a high score.
5. Turn the scores into probabilities $\hat{y} = softmax(Z) \in \mathbb{R}^{|V|}$.
6. We desire our probabilities generated, $\hat{y} \in \mathbb{R}^{|V|}$, to match the true probabilities, $y \in \mathbb{R}^{|V|}$, which also happens to be the one hot vector of the actual word.
The distance \( d(\hat{y}, y) \) is close to zero if \( \hat{y} \) is close to \( y \). Then, if the model is capable of finding \( \hat{y} \) such that \( d(\hat{y}, y) \) is close to the minimum (i.e. \( \hat{y} \approx y \)), that means that the model is very good at predicting the center word from the context. To learn the two vectors (the matrices \( B \) and \( A \), in general), CBOW defines a cost function that measures how good is the model at predicting the center word. Then, it optimizes the cost by updating the matrices \( B \) and \( A \) using stochastic gradient descent (SGD). The objective function to minimize is defined as

\[
J = -\log \mathbb{P}(w_c \mid w_{c-1}, w_{c+1}, \ldots, w_{c+m}) = -\log \mathbb{P}(b \mid \hat{a}) = -\log \frac{\exp(b^T \hat{a})}{\sum_{j=1}^{\vert V \vert} (b_j^T \hat{a})} = -b_c^T \hat{a} + \log \sum_{j=1}^{\vert V \vert} (b_j^T \hat{a})
\]

**Skip-Gram Model**

Skip-Gram Model is a model that predicts the surrounding word from the center word (context) [67]. In the CBOW approach, given an input sentence “The cat jump over the puddle” is to treat word “jumped” as the center word and predict or generate the surrounding context word “The”, “cat”, “over”, “the”, “puddle”.

The setup of skip-gram model is similar to CBOW, except swapping the \( x \) and \( y \) i.e. \( x \) in the CBOW is now \( y \) and vice-versa. The input one hot vector (center word) will be represented by \( x \) (since there is only one). And the output vectors as \( y_j \). The metrics \( A \) and \( B \) are defined the same way as in CBOW.

The way the skip-gram models works can be explained using the following six steps:

1. Generate the one-hot input vector \( x \in \mathbb{R}^{\vert V \vert} \) of the center word.

2. Get the embedded word vector for the center word \( a_c = Ax \in \mathbb{R}^n \)

3. Generate a score vector \( z = Ba_c \).
4. Turn the score vector into probabilities, \( \hat{y} = \text{softmax}(z) \). Note that the probabilities of observing each context word are \( \hat{y}_{c-m}, \ldots, \hat{y}_{c-1}, \hat{y}_{c+1}, \ldots, \hat{y}_{c+m} \).

5. We desire our generated probability vector to match the true probabilities which are \( y_{c-m}, \ldots, y_{c-1}, y_{c+1}, \ldots, y_{c+m} \), the one hot vectors of the actual output.

As in CBOW, to learn the two vectors (the matrices \( A \) and \( B \)), the skip-gram model defines a cost function that measures how good is the model at predicting the surrounding context word. The objective function to be minimized is defined as

\[
J = -\log P(w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m} \mid w_c)
= -\log \prod_{j=0, j \neq m}^{2m} P(w_{c-m+j} \mid w_c)
= -\log \prod_{j=0, j \neq m}^{2m} P(u_{c-m+j} \mid a_c)
= -\log \prod_{j=0, j \neq m}^{2m} \frac{\exp(b_{c-m+j}^T a_c)}{\sum_{k=1}^{\left| \mathcal{V} \right|} \exp(b_k^T a_c)}
= -\sum_{j=0, j \neq m}^{2m} b_{c-m+j}^T a_c + 2m \log \sum_{k=1}^{\left| \mathcal{V} \right|} \exp(b_k^T a_c)
\]

With this objective function, the method computes the gradients with respect to the unknown parameters and at each iteration and update them via SGD.

The loss functions \( J \), for CBOW and skip-gram, are expensive to compute because of the Softmax normalization, where we sum over all the \( |\mathcal{V}| \) scores. Instead of looping over the entire vocabulary, one can sample several negative examples, as introduced by Mikolov et al. [66], from a noise distribution \( (P_n(w)) \) which probabilities match the ordering of the frequency of the vocabulary. The new objective function for observing the context word \( w_{c-m+j} \) given the center word \( w_c \) would for the skip-gram model be:

\[
J = -\log \sigma(b_{c-m+j}^T a_c) - \sum_{k=1}^{K} \log \sigma(-\tilde{b}_k^T a_c)
\]

(2.7)

In the above formula, \( \{\tilde{b} \mid k = 1 \ldots K\} \) are sampled from \( P_n(w) \).
For CBOW, our new objective function for observing the center word $b_c$ given the context vector $\hat{a}$ would be:

$$J = -\log \sigma(b_c^T \hat{a}) - \sum_{k=1}^{K} \log \sigma(-\tilde{b}_k^T \hat{a})$$  \hspace{1cm} (2.8)$$

### 2.2 Distance or Similarity Measures

Given a set of points in $\mathbb{R}^n$, a distance measure is a function $d(x, y)$ that returns a real number as a distance between two points $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$. The distance or similarity function satisfies the following axioms [48]:

1. $d(x, y)$ is non-negative.
2. $d(x, y) = 0$, if and only if $x = y$.
3. $d(x, y) = d(y, x)$.
4. $d(x, y) \leq d(x, z) + d(z, y)$ (the triangle inequality).

#### 2.2.1 Cosine Similarity

The cosine distance between two points is the angle between the corresponding vectors of those points, in the range $[0, 180]$ degrees, regardless of how many dimensions the space has. The cosine distance makes sense in spaces that have high dimensions, including Euclidean spaces and discrete versions of Euclidean spaces, such as spaces where points are vectors with integer or Boolean components [48]. We can calculate the cosine distance as

$$d(x = (x_1, x_2, \ldots x_n), y = (y_1, y_2, \ldots y_n)) = \frac{\sum_{i=0}^{n} x_i y_i}{\sqrt{\sum_{i=0}^{n} x_i^2} \sqrt{\sum_{i=0}^{n} y_i^2}}$$  \hspace{1cm} (2.9)$$
2.2.2 Euclidean Distance

An n-dimensional Euclidean space is one where points are vectors of \( n \) real numbers. The conventional distance measure in this space, which we shall refer to as the L2-norm, is defined as

\[
d(x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n)) = \sqrt{\sum_{i=0}^{n} (x_i - y_i)^2}
\] (2.10)

2.2.3 Manhattan Distance

There are other distance measures that have been used for Euclidean spaces. For any constant \( r \), we can define the \( L_r \)-norm, to be the distance measure, by

\[
d(x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n)) = \left( \sum_{i=0}^{n} \|x_i - y_i\|^{r} \right)^{\frac{1}{r}}
\] (2.11)

Another common distance measure, a specific case of \( L - r \)-norm is the \( L_1 \)-norm, or Manhattan distance, where the distance between two points is the sum of the magnitudes of the differences in each dimension.

2.2.4 Jaccard similarity

The Jaccard similarity of sets \( X \) and \( Y \) is the ratio of the size of the intersection of \( X \) and \( Y \) to the size of their union, defined as

\[
SIM(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}
\] (2.12)

2.3 Models Utilized in the Thesis

2.3.1 Locality Sensitive Hashing

Locality sensitive hashing (LSH) was proposed by Indyk and Motwani [83] and finds its application in different areas. LSH was first applied in indexing high-dimensional points.
for Hamming distance [35], and later extended to $L_p$ distances [24] where $L_2$ is Euclidean distance, sign-random projection for angle based projection [17].

A general idea behind LSH is to use certain hash functions and to hash each document for several times in a way that similar documents with very high similarity are hashed into the same “bucket”. The hash functions used to have the property that similar points have a higher probability to be mapped together than dissimilar points. Then, we consider each pair related to the same bucket as a candidate pair in each hash. Thus, it is enough to only consider the candidate pairs to find similar elements. The procedure for indexing documents in the data set $D$ using LSH is done as follows [83, 106]:

1. Select $k$ hash functions $h$ randomly and uniformly from the LSH hash function family $H$ and create $L$ buckets for each hash function.

2. Create a hash table by hashing all documents in the data set into different buckets based on their hash values.

The procedure for searching similar documents to a new document $d'$ is as follows (more detailed explanation can be found in [83, 106]):

1. When a new document $d'$ arrives, use the same set of $k$ hash functions $h$ to map $d'$ into $L$ buckets, one from each hash table.

2. Retrieve all documents $d$ from the $L$ buckets, collect them into a candidate set $C$, and remove duplicate points in $C$ if any.

3. For each document $d$ in $C$, compute its distance to $d'$.

4. Return the nearest neighbors with the smallest distance to $d'$.

In LSH, the probability that two documents $d_1$ and $d_2$ are hashed into the same bucket is proportional to their distance $u = \|d_1 - d_2\|^2$, as defined in [83, 106], and can be computed as

$$p^h(u) = Pr[h(d_1) = h(d_2)] = \int_0^w \left( \frac{1}{u} sf_s(t) \left( 1 - \frac{t}{w} \right) \right) dt$$  \hspace{1cm} (2.13)
where $f_s$ is the probability density function of the hash $H$ and $w$ is the bucket width. For any given $w$, this probability decreases as the distance $u$ increases. Using $p^s(u)$, we can further compute the collision probability, namely, the success probability under $H$ ([83]) as

$$p^c(u) = \Pr[H(d_1) = H(d_2)] = 1 - (1 - (p^s(u))^k)^L$$ (2.14)

The process of creating hash tables and searching similar items from the hash table when for new documents is presented in algorithm 1 and algorithm 2 respectively.

---

**Algorithm 1** Pre-processing

1: **Input:** set $D$ of documents, number $L$ of hash tables
2: **Output:** Hash tables $T_i$, $i = 1, \ldots, L$
3: for all $i \in 1, \ldots, L$ do
4: Initialize hash table $T_i$ by generating a random hash function $g_i(.)$
5: end for
6: for all $i \in 1, \ldots, L$ do
7: for all $d \in D$ do
8: Store document $d$ on bucket $g_i(d)$ of hash table $T_i$
9: end for
10: end for
11: Return the hash tables $T_1, T_2, \ldots, T_L$

**Algorithm 2** Similarity Search using Approximate Nearest Neighbor

1: **Input:** new document $d'$, number $K$ of approximate nearest neighbors, hash tables $T_i, i = 1, \ldots, L$ created in Algorithm 1
2: **Output:** $K$ approximate nearest neighbors
3: $S \leftarrow \emptyset$
4: for all $i \in 1, \ldots, L$ do
5: $S \leftarrow S \cup \{\text{documents found in } g_i(d') \text{ buckets of table } T_i\}$
6: end for
7: Return the $K$ nearest neighbors of $D'$ found in the set $S$

---

### 2.3.2 Word Movers Distance

Word mover’s distance (WMD) is derived from the Earth movers distance as a distance measure on two documents. WMD utilizes the property of word2vec embeddings where
text documents are represented as a weighted point clouds of embedded words. The distance between two text documents \( d_1 \) and \( d_2 \) is the minimum cumulative distance that words from document \( d_1 \) need to travel to exactly match the point cloud of document \( d_2 \).

Assume that we are provided with word2vec embedding matrix \( X \in \mathbb{R}^{m \times n} \) for a finite size vocabulary of \( n \) words (i.e. \( |V| = n \)) where the \( i \)th column, \( x_i \in \mathbb{R}^m \), represents the embedding of the \( i \)th word in an \( m \)-dimensional space.

Given the vector representation of each word from the document, the goal is to incorporate semantic similarity between pairs of individual words into the document distance metric. Such a measure of word dissimilarity is naturally provided by their Euclidean distance in the word2vec embedding space [56]. Mathematically, the distance between word \( w_i \) and word \( w_j \) can be computed as follows:

\[
c(w_i, w_j) = \|x_i - x_j\|^2 \tag{2.15}
\]

where \( x_i \) and \( x_j \) are the embedding’s of the words \( w_i \) and \( w_j \), respectively. Using equation 2.15 we can compute the so-called word travel cost or the distance between any two word vectors. Therefore, we can use the word travel cost to compute the distance between two documents.

Let \( d \) and \( d' \) be normalized bag of words (nBOW) representations of two documents \( d \) and \( d' \), respectively. Let \( T \in \mathbb{R}^{n \times n} \) be a flow matrix, where \( T_{ij} \geq 0 \) denotes how much the word \( w_i \) in \( d \) has to “travel” to reach the word \( w_j \) in \( d' \), and \( n \) is the number of unique words appearing in both \( d \) and \( d' \). To transform \( d \) to \( d' \) entirely, we ensure that the complete flow from the word \( w_i \) equals \( d_i \), i.e \( \sum_j T_{ij} = d_i \) and the incoming flow to the word \( w_j \) equals \( d'_j \), i.e \( \sum_i T_{ij} = d'_j \). Using the above assumptions, the distance between two documents can be computed as the minimum cumulative cost required to move all words from \( d \) to \( d' \), defined as

\[
\min_{T \geq 0} \sum_{i,j=1}^n T_{ij} c(w_i, w_j) \tag{2.16}
\]
Feature Representation and Models

subject to
\[ \sum_{j=1}^{n} T_{ij} = d_i, \forall i \in \{1, \ldots, n\}, \sum_{i=1}^{n} T_{ij} = d'_j, \forall j \in \{1, \ldots, n\} \]

The solution is achieved by finding \( T_{ij} \) that minimizes the expression in Equation 2.15. In [56], this was applied to document classification using \( k \)-NN which produced outstanding performance among other state-of-the-art approaches.

2.3.3 Latent Semantic Analysis

Latent semantic analysis (LSA) derives a high-dimensional vector representation based on analyses of large corpora [25]. LSA is a classical tool for automatic extraction of similarities between documents, through dimensionality reduction. A term-document matrix is filled with weights corresponding to the importance of the term in the specific document (term-frequency/inverted document frequency) and then is reduced via SVD to a lower-dimensional space called the concept space. Formally, the term-document matrix \( X \) of dimension \( n \times m \) (\( n \) terms and \( m \) documents) can be decomposed into \( U \) and \( V \) orthogonal matrices and \( \Sigma \) a diagonal matrix through singular value decomposition as

\[ X = U \Sigma V^T \quad (2.17) \]

where

- \( X \in \mathbb{R}^{n \times m} \) is the term-document matrix
- \( U \in \mathbb{R}^{n \times n} \) is the term-term orthogonal matrix having the left singular vectors of \( X \) as columns
- \( V \in \mathbb{R}^{m \times m} \) is the document-document orthogonal matrix having the right singular values of \( X \) as columns
- \( \Sigma \in \mathbb{R}^{m \times m} \) is a term-document diagonal matrix whose elements are the singular values of \( X \) (the non-negative square roots of the eigenvalues of \( XX^T \))
This, in turn can be represented through a rank $k$ approximation of $X$ in a smaller dimensional space ($\Sigma$ becomes a $k \times k$ matrix).

$$X_k = U_k \Sigma_k V_k^T$$

(2.18)

This truncation process provides the basis for generating a $k$-dimensional vector space. Both terms and documents are represented by $k$-dimensional vectors in this vector space. This representation is used to compare documents in this new space. The $U_k$ matrix of dimensions $n \times k$ represents the model of words in the new $k$-dimensional concept space. We can, thus, compare the relative similarity of each word by taking the cosine distance between their representations. LSA has been used both as a theoretical model and as a tool for the characterization of semantic relatedness of units of language (see [25] for a more details).

### 2.3.4 Latent Dirichlet Allocation

One of the primary goals in natural language processing is to analyse texts by its topics. The process of extracting such topics from a collection of documents is called topic modeling. A topic can be described as a set of words that usually occur in similar contexts. Hence, the main objective of topic modelling is to discover hidden or latent structure from a collection of documents that share the same content.

Latent Dirichlet allocation (LDA) is one of the most popular techniques in topic modeling which treats each documents as a mixture of topics [12]. In LDA, we assume that there are $k$ underlying latent topics according to which documents are generated and that each topic is represented as a multinomial distribution over the $|V|$ words in the vocabulary. A document is generated by sampling a mixture of these topics and then sampling words from that mixture [12]. In LDA the following concepts are used:

- A word is a unit-basis vector fetched from a vocabulary $V$. We represent words using unit-basis vectors that have a single component equal to one and all other
components equal to zero. Thus, the \( v \)th word in the vocabulary is represented by a “\(|V|\)-vector” \( w \in \mathbb{R}^{|V|} \) such that \( w_v = 1 \) and \( w_u = 0 \) for \( u \neq v \).

- A document is a sequence of \( n \) words denoted by \( d = (w_n)_{n \in \mathbb{N}} \), where \( w_i \) is the \( i \)th word in the sequence.

- A corpus is a collection of \( m \) documents denoted by \( D = \{d_1, d_2, \ldots, d_m\} \).

We wish to find a probabilistic model of a corpus that not only assigns high probability to members of the corpus, but also assigns high probability to other “similar” documents. The LDA defines the following generative process for each document \( d \) in a corpus \( D \):

1. Choose \( n \sim \text{Poisson}(\xi) \), where \( n \) is the length of the document

2. Choose \( \theta \sim \text{Dir}(\alpha) \), where \( \theta_i \) is the multinomial topic distribution for the document \( d_i \) (a \( k \)-vector lies in the \((k-1)\)-simplex if \( \theta_i \geq 0 \), \( \sum_{i=0}^{k} \theta_i = 1 \)) and \( \alpha \) is a \( k \)-vector (\( k \) denotes the number of topics) with components \( \alpha_k > 0 \). The probability density function of the Dirichlet distribution is defined as:

\[
p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1-1} \times \cdots \times \theta_k^{\alpha_k-1} \tag{2.19}
\]

3. For each of \( w_i \) (1 \( \leq i \leq n \)):
   - Choose a topic \( z_i \sim \text{Multinomial}(\theta) \);
   - Choose a word \( w_i \) from a multinomial probability \( p(w_i|z_i, \beta) \) conditioned on the topic \( z_i \). The word probabilities are set by a \( k \times |V| \) matrix \( \beta \) where \( \beta_{ij} = p(w_j = 1|z_l = 1) \);
   - Note: each topic has a different probability of generating each word

In LDA, we generate both word distribution \( \beta_k \) for each topic and topic distribution \( \theta_d \) for each document using Dirichlet distributions. For instance, by learning a \( k \)-vector \( \alpha_l \), we learn the topics distribution the document may have. The higher the \( \alpha_l \) value is relevant to others, the more likely we will pick that topic. Given the parameters \( \alpha \) and \( \beta \), the joint
distribution of a topic mixture $\theta$, a set of $k$ topics $Z = \{z_1, z_2, \ldots, z_k\}$, and a set of $n$ words $W = \{w_1, w_2, \ldots, w_n\}$ is given by:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta)$$

(2.20)

where $p(z_n|\theta)$ is simply $\theta_i$ for the unique $i$ such that $z_n^i = 1$. The LDA model is represented as a probabilistic graphical model in Figure 2.1. As the figure makes clear, there are three levels to the LDA representation. The parameters $\alpha$ and $\beta$ are corpus-level parameters, assumed to be sampled once in the process of generating a corpus. In a Dirichlet distribution, the probability distribution is a sampled value from $p$ which always sums up to one. This is a reason why Dirichlet distribution is called a distribution of distributions.

### 2.3.5 Clustering

Cluster analysis groups data objects based only on information found in the data that describes the objects and their relationships. The goal is that the objects within a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups. The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better or more distinct the clustering.
Clustering is the process of examining a collection of “points” and grouping the points into “clusters” according to some distance measure. The goal is that points in the same cluster have a small distance from one another, while points in different clusters are at a large distance from one another.

Therefore, in the context of utility, cluster analysis is the study of techniques for finding the most representative cluster prototypes [48].

1. Summarization: Many data analysis techniques, such as regression or PCA, have a time or space complexity of $O(m^2)$ or higher (where $m$ is the number of objects), and thus, are not practical for large data sets. However, instead of applying the algorithm to the entire data set, it can be applied to a reduced data set consisting only of cluster prototypes. Depending on the type of analysis, the number of prototypes and the accuracy with which the prototypes represent the data, the results can be comparable to those that would have been obtained if all the data could have been used.

2. Compression: Cluster prototypes can also be used for data compression. In particular, a table is created that consists of the prototypes for each cluster. This means that each prototype is assigned an integer value that is its position (index) in the table. Each object is represented by the index of the prototype associated with its cluster. This type of compression is known as vector quantization and is often applied to image, sound, and video data, where (1) many of the data objects are highly similar to one another, (2) some loss of information is acceptable, and (3) a substantial reduction in the data size is desired.

3. Efficiently Finding Nearest Neighbors: Finding nearest neighbors can require computing the pairwise distance between all points. Often, clusters and their cluster prototypes can be found much more efficiently. If objects are relatively close to the prototype of their cluster then we can use the prototypes to reduce the number of distance computations that are necessary to find the nearest neighbors of an object. Intuitively, if two cluster prototypes are far apart then the objects in the corresponding clusters cannot be nearest neighbors of each other. Consequently, to find an
object’s nearest neighbors it is only necessary to compute the distance to objects in nearby clusters, where the nearness of two clusters is measured by the distance between their prototypes.

The two major approaches to clustering – hierarchical and point-assignment.

1. Hierarchical or agglomerative algorithms start with each point in its own cluster. Clusters are combined based on their “closeness” using one of many possible definitions of “close”. Computation stops when further combination leads to clusters that are undesirable for one of several reasons. For example, we may stop when we have a predetermined number of clusters. Or, we may use a measure of compactness for clusters and refuse to construct a cluster, by combining two smaller clusters, if the resulting cluster has points that are spread out over a too large region [48].

2. The other class of algorithms involve point assignment. Points are considered in some order and each one is assigned to the cluster into which it best fits. This process is normally preceded by a short phase in which initial clusters are estimated. Variations allow occasional combining or splitting of clusters, or may allow points to be unassigned if they are outliers (points too far from any of the current clusters) [48].

Algorithms for clustering can also be distinguished by the following:

1. Whether the algorithm assumes a Euclidean space or whether the algorithm works for an arbitrary distance measure. We shall see that a key distinction is that in a Euclidean space it is possible to summarize a collection of points by their centroid – the average of the points. In a non-Euclidean space there is no notion of a centroid and we are forced to develop another way to summarize clusters.

2. Whether the algorithm assumes that the data is small enough to fit in main memory or whether data must reside in secondary memory. Algorithms for large amounts of data often must take shortcuts since it is infeasible to look at all pairs of points. It is also necessary to summarize clusters in main memory since we cannot hold all the points of all the clusters in main memory at the same time.
K-means Algorithms

The best known family of clustering algorithms of this type is called k-means. They assume a Euclidean space, and they also assume the number \( k \) of clusters known in advance. It is, however, possible to deduce \( k \) by trial and error. After an introduction to the family of \( k \)-means algorithms, we shall focus on a particular algorithm, called BFR, that enables us to execute \( k \)-means on data that is too large to fit in main memory [48].

There are several ways to select the initial \( k \) points that represent the clusters. The heart of the algorithm is the for-loop, in which we consider each point other than the \( k \) selected points and assign it to the closest cluster. “Closest” means closest to the centroid of the cluster. Note, that the centroid of a cluster can migrate as points are assigned to it. However, since only points near the cluster are likely to be assigned, the centroid tends not to move too much after some iterations.

Algorithm 3 Basic \( k \)-means algorithm

1: **Input:** Selected \( K \) points as initial centroids
2: **repeat**
3: Form \( K \) clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster using distance functions
5: **until** centroids do not change

Centroids and Objective Functions of \( k \)-means

The goal of clustering is typically expressed by an objective function that depends on the proximities of points to one another or to the cluster centroids, e.g. minimize the squared distance of each point to its closest centroid. We illustrate this with two examples. However, the key point is this: once we have specified a proximity measure and an objective function, the centroid that we should choose can often be determined mathematically.

An objective function, which measures the quality of a clustering, is the sum of the squared error (SSE), also known as scatter. In other words, we calculate the squared error of each data point, i.e. its Euclidean distance to the closest centroid, and then compute the total sum of these squared errors. Given two different sets of clusters that are produced by
two different runs of \( k \)-means, we prefer the one with the smallest squared error since this means that the prototypes (centroids) of this clustering are a better representation of the points in their cluster. The SSE is formally defined as

\[
SSE = \sum_{i=1}^{k} \sum_{x \in C_i} \text{dist}(c_i, x)^2
\]  

(2.21)

where \( \text{dist} \) is the standard Euclidean \((L_2)\) distance between two objects, \( C_i \) is the set of points in \( i \)th cluster and \( c_i \) is the center of the \( i \)th cluster. Given these assumptions, it can be shown that the centroid that minimizes the SSE of the cluster is the mean. The centroid (mean) of the \( i \)th cluster is defined as

\[
c_i = \frac{1}{m_i} \sum_{x \in C_i} x
\]  

(2.22)

The \( k \)-means algorithm directly attempts to minimize the SSE and forms clusters by assigning points to their nearest centroid, which minimizes the SSE for the given set of centroids. Then, it recomputes the centroids so as to further minimize the SSE. However, the actions of \( k \)-means are only guaranteed to find a local minimum with respect to the SSE since they are based on optimizing the SSE for specific choices of the centroids and clusters, rather than for all possible choices.
Chapter 3

Pair-wise: Automatic Short Essay Evaluation

3.1 Motivation

Automatic essay evaluation, also called Automated Essay Scoring, plays an important role in the educational process and scoring subjective type of questions is one of the most expensive and time-consuming activity for educational assessments. As a consequence, the interest and the development of automated assessment systems are growing. Automated short essay evaluation (ASEE) can be seen as a prediction problem, which automatically evaluates and scores essay solutions provided by students via computer programs [68]. For academic institutions, ASEE represents not only a tool to assess learning outcomes, but also helps to save time, effort and money without lowering the quality of teacher’s feedback on student solutions.

The area has been developing since the 1960s when Page and his colleagues [71] introduced the first ASEE system. Various kinds of algorithms, methods, and techniques have been proposed to implement ASEE solutions, however, most of the existing ASEE approaches consider text semantics very vaguely and focus mostly on its syntax.

We can assume that most of the existing essay scoring systems give much more focus on syntax, vocabulary and shallow content measurements and only limited concerns for
the semantics using supervised ML approach. This assumption follows from the fact that
the details of most of the known systems have not been released publicly. To semantically
analyze and evaluate documents in these systems, Latent Semantic Analysis (LSA) [25],
Latent Dirichlet Allocation (LDA) [12], Content Vector Analysis (CVA) [4] and Neural
Word Embedding (NWE) [67, 56] are mostly used. NWE [10, 67] is similar to other text
semantic similarity analysis methods such that LSA or LDA. Their main difference is that
LSA and LDA utilize co-occurrences of words while NWE learns to predict context using
recurrent neural networks. Moreover, training of semantic vectors is resulted from neural
networks. NWE models have increased acceptance in recent years because of their high
performance in natural language processing (NLP) tasks [60].

Essay scoring systems that use the classical ML approaches, like LSA, LDA and others,
ignore the order of words or arrangement of sentences in its analysis of the meaning of
a essay because it does not have such a feature. The essay in LSA is treated as a “bag of
words” – an unordered collection of words. As such, the meaning of an essay, as derived
by classical ML approaches, are not the same as that which could be understood by human
beings from grammar, syntactic relations, logic, or morphological analysis. The second
problem is that classical ML approaches do not deal with polysemy well. This is because
each word is represented in the semantic space as a single point and its meaning is the
average of all its different meanings in the corpus [26]. The third problem is, as most of
the state-of-the-art essay scoring systems rely on supervised ML approaches, it requires a
large number of manually labeled training essays to train the scoring engine. There are
cases in which it is challenging to find labelled “training” short answers that require a great
deal of effort when creating labelled sets.

Therefore, to address the issues outlined above, we proposed an unsupervised and
semantic-based essay scoring system, called Pair-wise essay scoring system (“Pair-wise”,
in short). The task of the Pair-wise approach is to predict the score of the essay given
the student answer and reference answer. Instead of using input representation based on
bag-of-words, Pair-wise considers both the student answer and reference as a sequence of
words with rich contextual structure, and it retains maximum contextual information in
its projected latent semantic representation by projecting each word in its context onto a low-dimensional continuous feature vector using skip-gram models [67]. Then, the Pair-wise method computes the semantic similarity between feature vector representation of the reference solution and a student answer using Relaxed word mover’s similarity (RWMS) which is a modified method utilizing the word movers distance [67, 56].

The main objective behind the proposed pair-wise approach is not to exactly reproduce the human grader’s scores, which are varying in their evaluation but to provide acceptable and reliable scores and also to provide immediate and helpful feedback to the students and, also, to show that essays can be evaluated using semantic features and unsupervised approaches as well. The proposed pair-wise approach was compared with approaches using LSA, Wordnet and lexical similarity. The performance of the algorithm was evaluated both qualitatively and quantitatively. A qualitative evaluation measure based on Pair-wise ranking, called prank, for assessing the performance of various models w.r.t. the human scoring is also proposed [88]. For quantitative evaluation, the normalized root mean squared error (nRMSE) between human and predicted scores were used.

The rest of this chapter is organized as follows: Section 3.2 reviews existing ASEE approaches. In Section 3.3, the proposed Pair-Wise ASEE approach is introduced. Experiments and results are described in Section 3.4. Section 3.5 presents the summary of the chapter.

3.2 Related Work

The research on automatically evaluating and scoring essays is ongoing for more than a decade where Machine Learning (ML), NLP and artificial neural networks (NN) were used for evaluating essay question answers. In this section, we present the characteristics of the majority of essay scoring systems developed by commercial organizations and also systems introduced by the academic community.

Project Essay Grade (PEG) was the first essay scoring system developed by Ellis Page and his colleagues [71]. It evaluates and scores essays by measuring trins and proxes. A
trin is defined as an intrinsic higher-level variable, such as punctuation, fluency, diction, grammar, the number of paragraphs, average sentence length, the length of essay in words, counts of other textual units, etc. which, as such, cannot be measured directly and has to be approximated by means of other measures, called proxes \[82, 101, 111\]. However, the exact set of textual features underlying each dimension and the details concerning the derivation of the overall score were not disclosed \[9, 82\]. The system uses regression methods to score new essays based on a training set.

Intelligent Essay Assessor (IEA) uses on LSA \[30\] to infer characteristics that define the content, organizational, and development-related attributes of the essay. IEA also extracts other essential features that measure lexical sophistication, grammatical, mechanical, stylistic and organizational aspects of essays using natural language processing techniques. The system uses different attributes to measure the above aspects in essays, such that, context (e.g. semantic similarity), lexical sophistication, grammar (grammatical errors and grammatical error types), mechanics (e.g. spelling, upper and lower case), style, organization and development (e.g. sentence/sentence coherence, general essay coherence and topic development). IEA requires training essay set with a representative sample of a manually annotated essay by experts.

IntelliMetric was designed and introduced in 1991 by Vantage Learning as proprietary essay scoring \[81\]. IntelliMetric analyzes the semantic, syntactic and discursive attributes of essays to form a composite sense of meaning. The attributes used in IntelliMetric can be grouped into two main groups (content and structure). The content attributes are used to evaluate the subject matter covered, the breadth of content, the support of advanced concepts, the logic of discourse, the cohesion and consistency of the purpose and main idea. The structural attributes evaluate spelling, punctuation, letter cases, grammar, syntactic literacy, syntactic diversity, sentence complexity, use, legibility, and subject-binding. The system uses multiple predictions with linear analysis, Bayesian approach and LSA in predicting the final score, and combines the models into a single final score.

E-Rater \[5\] works by extracting features such as grammatical errors, word usage errors, writing mechanics errors, presence of essay-based discourse elements, development of
essay-based discourse elements, style flaws, content vector analysis (CVA) to evaluate current word usage, an alternative, differentiated measurement of word usage, based on the relative frequency of a word in high-scoring versus low-scoring essays. Such an attribute that takes into account the correct use of prepositions and collocations, and diversity in the formation of sentence structures and finally apply regression modelling to predict the score.

SAGrader [13] is also a proprietary essay scoring system developed by IdeaWorks, Inc. SAGrader combines a number of linguistic, statistical and artificial intelligence approaches to evaluate the essay automatically. The operation of the SAGrader is as follows: The instructor first specifies a task in a prompt. Then the teacher creates a section in which he identifies the “desired properties”, i.e. vital elements of knowledge (facts) to be included in a right answer as well as the relationships between these elements via a semantic network. Fuzzy logic allows the program to recognize the features in students’ essays and compare them with the desired ones. Finally, an expert system evaluates student essays based on the similarities between the desired and observed characteristics. Students immediately receive feedback by reporting their results, along with detailed comments on what they have done well and what remains to be done. The system provides both results and meaningful feedback through ontology-based information extraction.

Islam and Hoque [44] developed an essay scoring system using generalized latent semantic analysis (GLSA), which develops a system that uses grams for document matrix instead of word for document matrix, as in LSA. The system uses the following steps in the grading procedure: Pre-processing of training essays, removal of stop words, word origin, selection of n-gram index terms, n-gram when creating document matrices, calculation of singular value decomposition (SVD) from n-gram to document matrix, dimensional reduction of SVD matrices and calculation of similarity evaluation. The main advantage of GLSA is adherence to the word order in sentences. In order to reduce memory and time consumption without lowering the performance of automated scoring in comparison with human scoring, [111] proposed incremental singular value decomposition as a part of incremental LSA to score essays when the dataset is massive.
An extension of existing essay scoring systems was introduced in [114] by incorporating additional semantic coherence and consistency attributes. They designed coherence attributes by transforming sequential parts of an essay into the semantic space and measuring changes between them to estimate coherence of the text and consistency attributes that detect semantic errors using information extraction and logic reasoning. The resulting system, named “sage-semantic automated grader for essays”, provides semantic feedback for the writer and achieves significantly higher grading accuracy compared with other state-of-the-art essay scoring systems.

An investigation on the effectiveness of using semantic vector representations for the task of essay scoring was performed in [46]. According to the evaluation results on the standard English dataset, the effectiveness brought by the proposed semantic representations of essays depends on the learning algorithms and the evaluation metrics used. On the other hand, the effectiveness of individual semantic features is stable with respect to different numbers of dimensions.

In [29], an essay scoring system based on n-gram and cosine similarity were described. N-Gram was used for feature extraction and modified to split by word instead of by letter so that the word order would be considered. Based on evaluation results, this system got the best correlation of 0.66 by using unigram on questions that do not consider the order of words in the answer. For questions that consider the order of the words in the answer, bi-gram has the best correlation value by 0.67.

The architecture of an essay scoring system based on a rubric, which combines automated scoring with human scoring, was proposed in [107]. The proposed rubric has five evaluation viewpoints: contents, structure, evidence, style, and skill as well as 25 evaluation items which are subdivided viewpoint. At first, the system automatically scores 11 items included in the style and skill such as sentence style, syntax, usage, readability, lexical richness, and so on. Then it predicts scores of style and skill from these items’ scores by multiple regression models. It also predicts contents’ score by the cosine similarity between topics and descriptions.
The C-Rater is a rule-based ASEE system first introduced by Leacock and Chodorow [58], and later expanded by Sukkarieh and Blackmore [84], where “C” stands for concept. In C-rater, a model of a candidate’s correct answer is created by a content expert and consists of a number of simple sentences, each of which represents a concept necessary for the candidate to answer correctly. An interface helps teachers to create their models, for example, by selecting suitable candidates from a list of automatically generated synonyms of a word. Both the teacher’s model and the learner’s response are transformed into a so-called canonical representation. They use, for example, co-reference resolution, spelling correction, and synonym substitution and bring the sentence into a canonical syntactic form by extracting predicate argument structures. The model is then automatically matched with the learner’s answer in a rule-based way, determining which parts of the answer represent a paraphrase of a sentence in the model.

Content Assessment Module (CAM) was introduced by Bailey and Meurers [6] as a short essay scoring system for the English CREE data. In their system, learners’ responses and target responses are aligned at different linguistic levels (tokens, chunks and dependency triples) using different types of equivalence proofs at the token level: Tokens that are identical on the surface, share the same lemma, are semantically close to each other, or are of the same semantic type can be aligned. Then chunk and dependency alignments are created based on these token alignments. The system also provides spell checking and pronoun resolution and uses pre-alignment filters to remove the punctuation and lexical material specified in the question. Several characteristics, such as the percentage of aligned tokens, chunks and dependency triples in both the target and learning response and the type of alignment are then used as input to a supervised classifier for ML.

Cutrone et al. [23] used NLP techniques to develop a system capable of automatically assessing short-answers based on the linguistic features of student response. The system reduces the supplied answer as well as the student response to their canonical form through a comprehensive text pre-processing phase. All words in the canonical form are tagged based on their part of speech. The student response and the supplied answer are then compared. In this comparison, features encapsulated within Word Net are utilized to ensure
that exact word matches are not necessary for determining the level of equivalence between the student response and the supplied answer.

Thanawala et al. [94] used an ontology-based tool for automatic evaluation for free-text short responses submitted by users in learning management systems (LMS) based discussion forums, or community-based question-answer forums. Their architecture is based on simple NLP techniques, WordNet and hand-coded logic to make sense of user questions and submitted responses using the semantic web ontology language (OWL). The experimental results show that their approach, which is tested on computer science subject operating systems, can be effectively used to evaluate the response by providing a score that a learner uses to satisfy the learner’s objective.

Mohler and Mihalcea [70] proposed a form of unsupervised ASEE, with the intuition that the more similar response is to the corresponding target response, the better the students’ response. They use text-to-text similarity measures to assess the similarity between learner and target answer on a corpus of computer science questions. They assess the quality of their method by calculating the correlation between the normalized individual similarity scores and the gold standard score of the response (which is annotated on a scale of 0.0 to 5.0 points). They use both corpus-based and knowledge-based similarity measures. They used LSA and explicit semantic analysis. They derive text-to-text similarity by aggregating the cosine word similarities: For each content word in the learner’s answer, they determine the similarity of the most similar word in the target answer and return the sum of these individual similarities, normalized according to the answer length, as an overall similarity score. They find that several similarity measures work well while the best performance is achieved with LSA trained on a domain-specific corpus. This model works well and they report a correlation of 0.50 between their scores and human annotations for the best performing configuration.

Essay scoring systems that use classical ML approaches ignore the order of words or arrangement of sentences in its analysis of the meaning of a essay, because it does not have such a feature. The essay in LSA is treated as a “bag of words” – an unordered collection of words. As such, the meaning of an essay as derived by classical ML approaches are not
the same as that could be understood by human beings from grammatical, syntactic relations, logic, or morphological analysis. The second problem is that classical ML approaches do not deal with polysemy well. This is because each word is represented in the semantic space as a single point and its meaning is the average of all its different meanings in the corpus [26]. The third problem is, as most of the state-of-the-art essay scoring systems rely on supervised ML approaches, it requires a large number of manually labeled training essay set to train the scoring engine. There are cases in which it is challenging to find labelled training short answers that require a great deal of effort when creating labelled sets.

Therefore, we proposed an unsupervised and semantically based short essay scoring system where the task is to predict the score based on the student’s response and the reference answer. Rather than using an input representation based on bag-of-words, our approach considers both the student’s response and the reference answer as a sequence of words with a rich contextual structure and maintains a maximum of contextual information in its projected latent semantic representation by projecting each word in its context onto a low-dimensional continuous feature vector using skip-gram models [67]. Then the Pair-wise method calculates the semantic similarity between the feature vector representation of the reference solution and a student response using the Relaxed Word Mover’s Similarity (RWMS) method[67, 56].

### 3.3 The Pair-Wise ASEE Approach

The most common way of computing similarity between two text documents is to have the centroids of their word embedding and evaluate an inner product between these two centroids [67, 56]. However, taking simple centroids of two documents is not a good approximation for calculating the distance between these two documents [56]. The distance between individual words in a pair of documents is measured as opposed to the average distance between the two documents. Therefore, the Word Mover’s Distance (WMD) that
calculates the minimum cumulative distance that words from one document need to travel to match words from the other document.

Let \( d \) and \( d' \) be nBOW representations of two documents \( d \) and \( d' \), respectively. Let \( T \in \mathbb{R}^{n \times n} \) be a flow matrix, where \( T_{ij} \geq 0 \) denotes how much the word \( w_i \) in \( d \) has to “travel” to reach the word \( w_j \) in \( d' \), and \( n \) is the number of unique words appearing in \( d \) and \( d' \). To transform \( d \) to \( d' \) entirely, we ensure that the complete flow from the word \( w_i \) equals \( d_i \), i.e. \( \sum_j T_{ij} = d_i \) and the incoming flow to the word \( w_j \) equals \( d'_j \), i.e. \( \sum_i T_{ij} = d'_j \). Using the above assumptions, the distance between two documents can be defined as the minimum cumulative cost required to move all words from \( d \) to \( d' \) and can be defined as

\[
\min_{T \geq 0} \sum_{i,j=1}^{n} T_{ij} c(w_i, w_j)
\]

subject to

\[
\sum_{j=1}^{n} T_{ij} = d_i, \forall i \in \{1, \ldots, n\}, \sum_{i=1}^{n} T_{ij} = d'_j, \forall j \in \{1, \ldots, n\}
\]

where \( c(w_i, w_j) \) refers to the “similarity” of words \( w_i \) and \( w_j \) (defined as, for example, the cosine similarity in the Equation 5.1).

The solution is achieved by finding \( T_{ij} \) that minimizes the expression in Equation 3.1. In [56], this was applied to obtain nearest neighbors for document classification, i.e. \( k \)-NN classification, which produced outstanding performance among other state-of-the-art approaches. Therefore, WMD is a good choice for semantic evaluation of the similarity between documents. The features of WMD can be used to semantically score a pair of texts such that, for example, student’s answers and reference solutions. WMD allows each word embedding to be partially aligned (travel) to multiple word embeddings of the other text, which requires solving a linear program and is too slow for our purposes. A more faster result can be obtained by relaxing the WMD optimization problem and removing one of the two constraints (in the Equation 3.1) [56]. In our work, we used a modified version of WMD, called relaxed word mover’s similarity (RWMS), for automatic essay scoring as follows:
First, compute the word travel cost using cosine similarity: The objective here is to learn and include semantic similarity between individual word pairs that exist in the input pairs of documents (the student’s answer, denoted here by \( Sa \), and the reference or candidate answer, denoted here by \( Ca \)) rather than their lexical similarity. Let \( Sa \) and \( Ca \) be nBOW representations of \( Sa \) and \( Ca \), respectively. The word embedding model is trained on a set of documents. Since the goal is to measure a similarity between \( Sa \) and \( Ca \), \( c(w_i, w_j) \) is redefined as a cosine similarity [88]

\[
c(w_i, w_j) = \frac{x_i x_j}{\|x_i\| \|x_j\|}
\]

where \( w_i \) and \( w_j \) are words from \( Sa \) and \( Ca \) respectively and \( x_i \) and \( x_j \) are the vector representations for words \( w_i \) and \( w_j \), respectively. Much tighter bound results can be obtained by relaxing the WMD optimization problem and removing one of the two constraints (Equation 3.1) and, use cosine similarity (Equation 3.2) instead of the Euclidean distance (Equation 2.15). The word mover’s distance defined in Equation 3.3 is also modified and redefined to a relaxed word mover’s similarity as

\[
\max_{T \geq 0} \sum_{i,j=1}^{n} T_{ij} c(w_i, w_j)
\]

subject to

\[
\sum_{j=1}^{n} T_{ij} = d_i, \forall i \in \{1, \ldots, n\}
\]

### 3.3.1 A Layered based Pair-Wise Architecture

The proposed Pair-Wise architecture is illustrated in Figure 3.1 which can be summarized as follows (detailed descriptions of its layers will be introduced in the following subsections):

The first layer is an input layer where both the candidate answer and the student answer are provided to the algorithm as an input. These inputs will be sent to the second layer for text pre-processing and the following activities will be carried out: Tokenization; Removing punctuation marks, determiners, and prepositions; Transformation to lowercase; Stop word removal [79] and Word stemming [73]. Every user input has to pass
through a proper validation mechanism before computing the score to increase the usability and trustworthiness of essay scoring systems. Therefore, in the third layer, shingle based plagiarism (see section 3.3.2) and foolish detection tasks (see section 3.3.3) are carried out. Those essays which are valid will be sent to the fourth layer for scoring and those who failed to pass the validation will automatically get a score of zero. The fourth layer is where the semantic similarity between the candidate answer and student answer is computed using relaxed word mover’s similarity (see section 3.3) to predict the score. The result is forwarded to layer five for storage. The most popular word embedding model proposed by Mikolov et al. [67], trained using the skip-gram method by maximizing the average log probability of all words, is used in this work.

Figure 3.1 The architecture of the proposed Pair-Wise ASEE approach.
3.3.2 Plagiarism detection using shingles

Plagiarism is one of the growing problems in the academic and research field, which is raising more and more concerns in the educational systems. When submitting solutions for essay tests using online systems, students may commit plagiarism by copying and pasting solutions from other students. It has become a very common problem in evaluating the work of students and their creativity.

In order to address this issue, we proposed a plagiarism detection approach that identifies duplicates within the submitted solutions by comparing one student’s answer with other student’s answers for the specific essays based on the $k$-shingles [74] approach. The most effective way to represent a document, for the purpose of identifying lexically similar documents, is to construct from the document a set of short strings that appear within it using shingles [74]. If we do so, then documents that share pieces as short as sentences or even phrases will have many common elements in their sets, even if those sentences appear in different orders in the two documents. By defining a $k$-shingle for a document to be any sub-string of length $k$ found within the document, we can associate with each document the set of $k$-shingles that appear one or more times within that document.

As the objective is to identify plagiarised answers using lexical similarity, we keep punctuation and white spaces in the process. This way of document representation keeps the word order and allows comparing documents based on the sets of character shingles. The similarity of two documents can be defined as the Jaccard similarity of the two sets of shingles, i.e. the number of elements (shingles) they have in common, as a proportion of the combined size of the two sets or the size of the intersection divided by the size of the union of these sets. The Jaccard similarity between two sets $X$ and $Y$ is defined as

$$S_j(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$ (3.4)

3.3.3 Foolish detection

Another issue in essay scoring is that it might be easily tricked or fooled by the students. One of the fooling methods that students might use for tricking the system, that is getting
higher scores, is just to list high profile words related to the topic and submit those words as a solution to the essay. In order to penalize such attempts, we implemented a simple foolish detection algorithm in two ways: The first method is done by computing the similarity between the student’s answer with text pre-processing and without text pre-processing. If the student answer contains only high profiled words, then the similarity between the two pairs will be almost the same. Therefore, the student is trying to fool the system and the system will automatically discard the student’s solution. The second method first generates content bearing words or high-profile words from the candidate answer using inverse document frequency (IDF). Then, similarity is computed between the student’s answer without any text pre-processing and content bearing words. If the similarity between the two is greater than or equal to some threshold level then the student is trying to fool the system and the system will also automatically discard the student’s answer.

3.4 Experiment

The experiment was carried out on datasets provided by the Hewlett Foundation at a Kaggle\(^1\) competition for ASEE. There are 10 datasets containing student essays from grade ten students. All the datasets were rated by two human raters. The features of the datasets are shown in Table 3.1. Five datasets, numbered 1, 2, 5, 9 and 10 were provided with the correct or reference answer to which student answers will be compared to. In case of the other five datasets (no. 3, 4, 6, 7 and 8) the reference answer was created according to the score given by human raters such that ten students’ answers which got full score were randomly selected as reference answer. We tokenized the essays using the NLTK\(^2\) tokenizer, lowercase the text, and normalize the gold-standard scores, provided by the human raters, to the range of \([0, 1]\). To learn the representation of each essay, the freely available word2vec\(^3\) word embedding model was used with an embedding for 3 million words/phrases from Google News, trained using the approach in [67]. Additionally, the

\(^1\)https://www.kaggle.com/c/asap-sas
\(^2\)http://www.nltk.org
\(^3\)https://code.google.com/archive/p/word2vec/
Scikit-learn⁴ and the Numpy⁵ Python libraries were used. The performance of the proposed Pair-Wise approach is compared to other three approaches utilizing LSA [25, 44], Wordnet [3, 98, 113] and cosine similarity [28, 105].

<table>
<thead>
<tr>
<th>Essay Set</th>
<th>Grade Level</th>
<th>Domain</th>
<th>Score range</th>
<th>Average length in words</th>
<th>Training set size</th>
<th>Test set size</th>
<th>Total size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>Science</td>
<td>0-3</td>
<td>50</td>
<td>1672</td>
<td>558</td>
<td>2230</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>Science</td>
<td>0-3</td>
<td>50</td>
<td>1278</td>
<td>426</td>
<td>1704</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>English, arts</td>
<td>0-2</td>
<td>50</td>
<td>1891</td>
<td>631</td>
<td>2522</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>English, arts</td>
<td>0-2</td>
<td>50</td>
<td>1738</td>
<td>580</td>
<td>2318</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>Biology</td>
<td>0-3</td>
<td>60</td>
<td>1795</td>
<td>599</td>
<td>2394</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>Biology</td>
<td>0-3</td>
<td>50</td>
<td>1797</td>
<td>599</td>
<td>2396</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>English</td>
<td>0-2</td>
<td>60</td>
<td>1799</td>
<td>601</td>
<td>2400</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>English</td>
<td>0-2</td>
<td>60</td>
<td>1799</td>
<td>601</td>
<td>2400</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>Science</td>
<td>0-2</td>
<td>60</td>
<td>1798</td>
<td>600</td>
<td>2398</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>Science</td>
<td>0-2</td>
<td>60</td>
<td>1799</td>
<td>599</td>
<td>2398</td>
</tr>
</tbody>
</table>

### 3.4.1 Quantitative Evaluation

Model validation in essay scoring systems depends on comparing the agreement between the predicted score of the model with the score given by the human raters [4]. In this scenario, the scores from human raters are considered as gold standard and function as an explicit criterion for evaluating the performance of the models being compared in the experiment.

The predicted score of each essay was compared with the human raters score to test the reliability of the proposed Pair-wise approach. Normalized root mean squared error (nRMSE) was used to measure the agreement between the score given by the Pair-Wise approach and the score given by the human raters. The essay scores provided by human raters were normalized to be within [0, 1]. nRMSE is one of the most widely used and

⁴http://scikit-learn.org/
⁵http://www.numpy.org/
accepted evaluation metric [103] and is defined as

\[
nRMSE(ES) = \left( \frac{\sum_{Sa \in ES} (r(Sa) - h(Sa))^2}{|ES|} \right)^{\frac{1}{2}}
\]

(3.5)

where \( ES \) is the essay set used, \( r(Sa) \) and \( h(Sa) \) are the predicted rating for the student answer \( Sa \) by the used ASEE approach and the human rating for \( Sa \), respectively. Rating here means how the student answer is similar to the reference solution. The lower the \( nRMSE \) the better the performance of the measured approach is.

Figure 3.2 A quantitative comparison using \( nRMSE \) (Equation 3.5) of the proposed Pair-Wise approach and the baselines using LSA, WordNet and Cosine similarity. In case of the datasets no. 3, 4, 6, 7 and 8, the average performance from 10 runs (corresponding to the 10 randomly chosen reference solutions) is reported while in case of the other five datasets (where only the one reference solution indicated in the data is used), the result from one run is reported.

![Graph showing nRMSE comparison](image)

Figure 3.2 shows the \( nRMSE \) between the human score and the tested ASEE systems for the datasets used in the experiment. Except the Dataset8, where Pair-Wise was performing
slightly worse than the winner Wordnet baseline, Pair-Wise was outperforming the baseline approaches.

Table 3.2 The p-values resulting from the Wilcoxon signed-rank test between the \(nRMSE\) results of the proposed ASEE and the baselines using LSA, WordNet and Cosine similarity.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Pair-Wise vs.</th>
<th>LSA vs.</th>
<th>Cosine vs. WordNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LSA</td>
<td>WordNet</td>
<td>Cosine</td>
</tr>
<tr>
<td>Dataset3</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>Dataset4</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>Dataset6</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>Dataset7</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
</tr>
<tr>
<td>Dataset8</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
</tr>
</tbody>
</table>

In case of the datasets 3, 4, 6, 7 and 8, the average values of \(nRMSE\) from the ten runs corresponding to ten randomly chosen reference solutions are indicated in the Figure 3.2. To test if the differences between the tested ASEE approaches indicated in the Figure 3.2, in case of these 5 datasets, are statistically significant, the non-parametric Wilcoxon signed-rank test was used [64]. Wilcoxon signed ranks test is a nonparametric statistical procedure that is often used in ML experiments to compare two sets of scores coming from the same methods without assuming normality in the data [42, 64]. The resulting p-values from these tests are reported in the Table 3.2 showing that the differences between Pair-Wise and the baselines as well as between the baselines are statistically significant.

### 3.4.2 Qualitative Evaluation

\(nRMSE\) measures the performance of the tested ASEE approaches quantitatively, that is, by how much the predicted score of an approach differs from the human rating(s). Since the proposed and baseline approaches are utilizing different models, their results might be biased. Thus, a qualitative evaluation measure, named \(prank\), referring to “Pair-wise ranking” was proposed by us defined as

\[
prank(ES) = \frac{1}{Z} \sum_{S_{a_i}, S_{a_j} \in ES} \delta(S_{a_i}, S_{a_j})
\]  

(3.6)
where $ES$ is the essay set, as introduced above, $Z = |ES|(|ES| - 1)/2$ is a normalization constant and $\delta(Sa_i, Sa_j) = 1$ if \((h(Sa_i) < h(Sa_j) \& r(Sa_i) < r(Sa_j))\) or \((h(Sa_i) > h(Sa_j) \& r(Sa_i) > r(Sa_j))\) while in cases where \(h(Sa_i) = h(Sa_j)\), $\delta(Sa_i, Sa_j) = 1 - |r(Sa_i) - r(Sa_j)|$.

In other words, $\delta(Sa_i, Sa_j)$ results in it’s maximal value 1 when the predicted ratings for two student answers $Sa_i$ and $Sa_j$ do not change the human “ranking” of $Sa_i$ and $Sa_j$ w.r.t. their similarities to the reference solution. If the human ranking can not be determined, i.e. the human rated the similarities of $Sa_i$ and $Sa_j$ to the reference solution equally, then the lower the difference between the predicted ratings the better.

Figure 3.3 A qualitative comparison using $prank$ (Equation 3.6) of the proposed Pair-Wise approach and the baselines using LSA, WordNet and cosine similarity. In case of the datasets no. 3, 4, 6, 7 and 8, the average performance from 10 runs (corresponding to the 10 randomly chosen reference solutions) is reported while in case of the other five datasets (where only the one reference solution indicated in the data is used), the result from one run is reported.
Figure 3.3 shows the results when measuring the (average) performance of the discussed approaches qualitatively using the proposed _prank_ measure. Pair-Wise outperforms the baselines in 7 from the 10 essay sets used for evaluation. In 2 cases, the _prank_ score was very close to the winner approaches while only in one case the proposed approach was substantially outperformed by the LSA baseline.

Table 3.3 The p-values resulting from the Wilcoxon signed-rank test between the _prank_ results of the proposed ASEE and the baselines using LSA, WordNet and Cosine similarity.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset3</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
<td>0.878</td>
<td>0.878</td>
<td>0.241</td>
<td></td>
</tr>
<tr>
<td>Dataset4</td>
<td>0.012</td>
<td>0.005</td>
<td>0.053</td>
<td>0.012</td>
<td>0.078</td>
<td>0.170</td>
<td></td>
</tr>
<tr>
<td>Dataset6</td>
<td>0.006</td>
<td>0.005</td>
<td>0.028</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
<td></td>
</tr>
<tr>
<td>Dataset7</td>
<td>0.016</td>
<td>0.005</td>
<td>0.005</td>
<td>0.053</td>
<td>0.721</td>
<td>0.006</td>
<td></td>
</tr>
<tr>
<td>Dataset8</td>
<td>0.005</td>
<td>0.005</td>
<td>0.005</td>
<td>0.332</td>
<td>0.044</td>
<td>0.006</td>
<td></td>
</tr>
</tbody>
</table>

In case of the datasets 3, 4, 6, 7 and 8, the average values of _prank_ from the ten runs corresponding to ten randomly chosen reference solutions are indicated in the Figure 3.3. To test if the differences between the tested ASEE approaches indicated in the Figure 3.3, in case of these 5 datasets, are statistically significant, the non-parametric Wilcoxon signed-rank test was used, as in the case of quantitative evaluation described above. The resulting p-values from these tests are reported in Table 3.3 showing that the differences between Pair-Wise and the baselines are statistically significant.

### 3.4.3 Comparison with the Other Systems

We also compared the Pairwise system with the results obtained from the leaderboard of the Hewlett Foundation⁶ short for answer scoring competition and with other systems from the literature who used the same data sets. In Table 3.4, we have ranked the leading systems from the competition together with the Pairwise system and systems from the literature. As the results are reported in terms of average accuracy across all data sets, we follow the same reporting mechanism to ensure fair comparisons. All reported systems use

---

⁶[https://www.kaggle.com/c/asap-sas/leaderboard](https://www.kaggle.com/c/asap-sas/leaderboard)
Table 3.4 Accuracy comparison of various systems from the Kaggle competition and the literature.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Systems</th>
<th>Avg. Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pairwise</td>
<td>0.831</td>
</tr>
<tr>
<td>2</td>
<td>Higgins et al. [38]</td>
<td>0.789</td>
</tr>
<tr>
<td>3</td>
<td>Ramachandran et al. [75]</td>
<td>0.780</td>
</tr>
<tr>
<td>4</td>
<td>L.Tandalla [85]</td>
<td>0.771</td>
</tr>
<tr>
<td>5</td>
<td>J.Zbontar [109]</td>
<td>0.771</td>
</tr>
<tr>
<td>6</td>
<td>ETS [47]</td>
<td>0.768</td>
</tr>
<tr>
<td>7</td>
<td>Measurement Inc. [49]</td>
<td>0.766</td>
</tr>
<tr>
<td>8</td>
<td>Stefan et al. [33]</td>
<td>0.763</td>
</tr>
<tr>
<td>9</td>
<td>Charlie [86]</td>
<td>0.761</td>
</tr>
<tr>
<td>10</td>
<td>James [45]</td>
<td>0.760</td>
</tr>
<tr>
<td>11</td>
<td>Conort [22]</td>
<td>0.757</td>
</tr>
<tr>
<td>12</td>
<td>Riordan et al. [76]</td>
<td>0.723</td>
</tr>
<tr>
<td>13</td>
<td>Horbach [40]</td>
<td>0.635</td>
</tr>
</tbody>
</table>

supervised machine learning except the works of Horbach [40] which is semi-supervised, while our system uses an unsupervised approach. The results show that our approach has outperformed traditional supervised approaches. We were not able to compare the Pairwise system with other systems in the literature, as most systems used their own data sets and some other different data sets.

### 3.4.4 Plagiarism Detection Evaluation

The dataset used for this task was provided by Clough [19] and was collected from students in Sheffield University studying for a degree in Computer Science at either undergraduate or postgraduate level. The dataset is publicly available. The dataset contains 100 students answers from which 95 answers were provided by the 19 participants and the 5 Wikipedia source articles for five learning tasks. For each learning task, there are 19 examples of each of the heavy revision, light revision and near copy levels and 38 non-plagiarized examples written independently from the Wikipedia source. The answer texts contain 19559 words in total while the Wikipedia pages contain 14242 words in total. The average length of files in the corpus is 208 words and 113 unique tokens. The metrics used to check the

7https://ir.shef.ac.uk/cloughie/resources/plagiarism_corpus.html
Pair-wise : Automatic Short Essay Evaluation

The performance of shingle based plagiarism detection are Precision, Recall and Accuracy defined in equations 5.3, 5.4 and 5.5, respectively as

$$\text{Precision} = \frac{TP}{TP + FP}$$ \hspace{1cm} (3.7)

$$\text{Recall} = \frac{TP}{TP + FN}$$ \hspace{1cm} (3.8)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$ \hspace{1cm} (3.9)

where True Positives (TP) is the number of documents actually plagiarized and predicted as plagiarized, False Positives (FP) is the number of documents predicted as plagiarized but actually are non-plagiarized, True negatives (TN) is the number of documents actually non-plagiarized and predicted as non-plagiarized and False Negatives (FN) are the number of documents actually plagiarized but predicted as non-plagiarized.

The challenge faced in the evaluation of “as plagiarized” or not is to set a minimum threshold level. We were unable to find any baseline used so far for this purpose, and it’s not actually easy to say that students’ solution is plagiarized or not because sometimes students might use and read the same reference materials. With the assumption that the students might use the same reference material and as they are asked the same type of questions, there might be some level of similarity in their solutions. Therefore, we set the similarity threshold value to 5%, 10%, 15% and 20% to get a clear separating line between actually plagiarized and predicted plagiarized documents.

Table 3.5 Shingle based plagiarism detection results

<table>
<thead>
<tr>
<th>Metrics</th>
<th>5%</th>
<th>10%</th>
<th>15%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.62</td>
<td>0.78</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Recall</td>
<td>1.00</td>
<td>0.96</td>
<td>0.94</td>
<td>0.77</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.62</td>
<td>0.82</td>
<td>0.96</td>
<td>0.86</td>
</tr>
</tbody>
</table>

The results of shingle based plagiarism detection is reported in table 3.5 using four different similarity threshold values (5%, 10%, 15% and 20%). The experiment showed
that as the threshold value increases from 5% to 10% and from 10% to 15%, the overall accuracy is increasing. But when the threshold value increases from 15% to 20%, the overall accuracy starts to decrease. Therefore, we can deduce that the minimum threshold level that can be used to detect plagiarized student solutions can be 15% for this specific case. In other words, 15% of similarity between students solution can be tolerable and will not be considered as plagiarized.

3.5 Summary

In this chapter, we presented an unsupervised semantic-based essay scoring system called the “Pair-wise” approach. The objective of Pair-wise is to predict the score of the essay given the student answer and reference answer. Instead of using input representation based on bag-of-words, Pair-wise considers both the student answer and reference as a sequence of words with rich contextual structure. It retains maximum contextual information in its projected latent semantic representation by projecting each word in its context onto a low-dimensional continuous feature vector using skip-gram models. Then, the Pair-wise computes the semantic similarity between feature vector representation of the reference solution and a student answer using the relaxed word mover’s similarity (RWMS) method which is a modified method from word movers distance. The performance of Pair-wise was evaluated both by a qualitative accuracy measure and qualitative accuracy measures, proposed by us for the purpose of deeper evaluation. From the experimental results, we can deduce that the proposed Pair-wise approach can be used in evaluating and scoring essay questions using semantic features and opens the opportunity in scoring essay exams using unsupervised ML techniques. ASEE systems with such features can be integrated with massive open online course (MOOC) systems to give automatic feedback to the students and, as the level of awareness increases, the number of participants in using MOOC will also increase.
Chapter 4

Reducing Annotation Effort

4.1 Motivation

Automatic short essay evaluation (ASEE) systems were introduced to alleviate the workload of the assessors and to improve the feedback cycle in the teaching-learning process in the context of large-scale assessment. Since its introduction, several research activities have been carried out [69]. The task of ASEE was regarded as a machine learning problem that learns to approximate the assessment process using handcrafted features with supervised machine learning approaches.

Most state-of-the-art ASEE systems rely on supervised machine learning approaches that require a large number of manually annotated training sets to train the scoring engine and make the scoring engine work well. There are cases in which it is difficult to find labeled “training” short essays that require a great deal of effort when creating labeled sets. For the evaluation of high-stakes short essays, the effort invested in manual marking of a large number of short essays to train a good model may be justified [110]. However, it becomes difficult in situations where new responses are generated more frequently because it is difficult to decide which and how many of the newly generated short essays (from the students) need to be manually labeled by the teacher to form a reliable rating machine.

In the classical approach of supervised machine learning, human annotators evaluate responses until a certain amount is reached that is considered adequate for the training of the
Reducing Annotation Effort

model. With manual annotation, the instances to be scored are selected at random. There is a problem, however, in such process: Annotators will inevitably see many similar answers that will not add much new knowledge to the trained model and the class distribution in the data is often highly distorted [110]. Researches that investigate on how to select instances to be annotated manually and, also, ways on how to reduce the number of annotated training examples required to train a model must be done to make ASEE systems more useful. The investigation on ASEE should, therefore, focus on how to replace the random selection of responses to be annotated with a more informal and convenient approach.

The most recent work on minimizing human annotation for short essay evaluation follows two strategies. The first approach is grouping students’ responses so that each set can be graded together [15, 110, 41]. The rationale behind using clustering is that similar essays will end up in the same cluster after clustering. The annotator (assessor) will only label (score) the prototype of each cluster and the label (score) will be then propagated to all members of the cluster. The second approach is using active learning [110] for selecting the most informative responses. The manual workload is reduced either to help the teachers directly or to support the creation of training data for automatic grading.

Little work has been done in investigating the extent to which the ASEE performance depends on the availability of training data and what proportion of short essays should be scored by the teacher manually (in case of no labeled short essay scenario). In this chapter, we propose an efficient way of selecting a small-sized training set to be annotated by human rater that will be used to train the scoring engine using semantic intelligent $k$-means and semantic locality sensitive hashing (Sem-LSH) techniques. Specifically, we investigate the question of what proportion of the short essays should the teacher score manually before the ASEE engine is used on new short essays without labels (scores) and how to select those responses proportionally. Thus, in this chapter, we introduce a new way of addressing the issue of reducing the workload of human graders in creating training sets for supervised ASEE.

The rest of this chapter is organized as follows: Section 4.2 provides an overview of the existing works and approaches. In Section 4.3.4, the proposed Sem-LSH approach
Reducing Annotation Effort is introduced. Section 4.4 presents experimental setting and data sets used. Results and discussion are presented in Section 4.5. Finally, Section 4.6 presents the summary of the chapter.

4.2 Related Work

In this section, we present the recent works in addressing the question of how to reduce the workload for human graders in creating training data for short essay scoring systems. We also show the comparison of supervised machine learning approaches with and without clustering approach where a subset of data points are selected for manual labeling as shown in figures 4.1 and 4.2 [110].

Figure 4.1 The classical supervised scoring approach where a subset of instances are selected for manual annotation and then training the model.

Figure 4.2 The classical supervised scoring approach with a clustering where a subset of instances from each cluster is selected proportionally for manual annotation then training the model.

The work most closely related to ours is Zesch et al. [110], which includes experiments with a form of sample selection based on the output of the clustering methods. The work of Horbach et al. [41] examines approaches to selecting the optimal number of short essays to be evaluated by a human grader and later used for training the scoring engine with clustering methods. After the human annotator has marked the optimal number of responses from each cluster, their approach propagates the human score to the rest of the cluster members. Their experiment was conducted with very short German responses. According
to their results, a scoring accuracy of about 85% can be achieved by annotating 40% of the training data.

Brooks et al. [15] proposed a more similar approach to Horbach et al. [41], which uses clustering to group student responses into groups and subgroups. It allows teachers to access these groupings so that they can read, rate and give feedback on a large number of responses at once. In their work, they reported that the use of clustering could save about 66% time.

Basu et al. [7] also used clustering approaches that automatically find groupings and subgroups of similar answers for the same question using \( k \)-Medoid and LDA. Once the groups are identified, the teacher can mark entire groups as right or wrong and give rich feedback to an entire group at once. When a label is assigned to a complete cluster, a teacher selects the label that matches most data.

Zesch et al. [110] conducted a sample selection experiment based on the output of clustering methods. By automatically bundling the essays, the elements located near their centroids are labelled and added to the training data. The motivation behind this approach is that elements with similar lexical material are expressed by similar characteristics, often have the same meaning and, in such cases, often deserve the same rating. By training in lexically different instances, the classifier should learn more than by training in very similar instances.

Horbach and Palmer [40] do active learning whereby selecting instances to be labelled manually to optimize the classifier’s performance quickly. They find that sample selection based on uncertainty is more efficient in improving the classifier than a random and a cluster-based baseline. Even if there are attempts to address in reducing efforts required in short essays, there is still much work to be done in the domain. Therefore, we used semantic locality sensitive hashing (Sem-LSH) in this work where the inputs to Sem-LSH are selected using semantic intelligent \( k \)-means. It is called intelligent since it is able to cluster kernel matrix without any information regarding to the number of required clusters [36]. Since the introduction, LSH has attracted communities from computer vision [54],
Reducing Annotation Effort

machine learning [63, 51, 112], natural language processing [43, 99, 100, 37, 92] and so on.

Therefore, in this work, we used semantic intelligent \( k \)-Means and semantic LSH to addressing the problem of reducing the workload of manually annotating while training automatic short essay scoring. We investigate the possibility of selecting a small proportion of training sets to be labeled, which will result in comparable performance with scoring engines trained on a large number of sets.

4.3 Proposed model

4.3.1 Problem statement

The problems that we tried to address are formulated as follows: The first problem is that given a set of unlabeled short essays \( U = \{u_1, u_2, ..., u_n\} \), the objective is to find an optimal way of scoring (labeling) short essays that can be comparatively sufficient to train the scoring engine so that the score distribution is well represented for each class and also as small as possible amount of representative answers are labelled. The second problem is, given a small set of scored short essays \( S_e \) and a large set of unlabeled short essays \( U_u \), efficiently find all points in \( S_e \) that are similar to each point in \( U_u \) using certain similarity or distance function.

4.3.2 Straightforward approach

The problem of creating a training dataset can be addressed using the traditional setting where human annotators evaluate responses until a specific number or score distribution is achieved that is considered sufficient for training the model. As long as the answers are selected randomly for manual evaluation, it is inevitable that annotators will see many similar answers that do not add much new knowledge to the trained model [110]. A further disadvantage is that the class distribution in the data is often strongly distorted. Thus, the number of responses that need to be evaluated manually is much higher than expected. It
should be possible to replace the random selection of responses to be annotated with a more informed approach.

The later problem of scoring can also be tackled in different ways with different approaches. Since we only have a small number of scored essays, the nearest neighbour’s approach might be an optimal one. The most common and easy way to find similar short essays, among other short essays in the collection is to compare all the short essays with each other. Nevertheless, as it is clear, comparing all pairs of short essays is time-consuming and not computationally efficient.

4.3.3 Efficient approach

The most efficient and practical approach for selecting short essays to be scored manually is by using an unsupervised machine learning approach called clustering. The clustering approach groups short essays into different sets so that the human annotator can score selected representative short essays from each set. In this work, we used a semantically enhanced clustering approach to consider both lexical and semantic diverseness of elements in the process of manual annotation. It maximizes the probability of class distribution in the data. Therefore, the training dataset will be more informative for the classifier or scoring engine.

The efficient approach for scoring new short essays when there is a small amount of scored short essays can be well addressed using hashing. Hashing is one of the most popular solutions for approximate nearest neighbour searches [43, 99]. In general, hashing is an approach of transforming a data element into a low-dimensional representation equivalent to a short code consisting of a sequence of bits. Using hashing to find nearest neighbours involves two options: indexing data elements using hash tables formed by storing elements with the same code in a hash bucket, and, approximating the distance using the so-called shortcodes. The first way considers elements belonging to the buckets that correspond to the codes of the new element as it’s nearest candidates using the advantage of the locality-sensitive property, i.e. that similar elements are more likely to be mapped to the same code rather than dissimilar elements. Therefore, we used the most efficient and widely
used hash-based approach, the locality sensitive hashing, which belongs to a family of randomized algorithms. This approach allows us to quickly find similar short essays from a large collection of short essays stored in the database [83, 106].

### 4.3.4 Semantic LSH (Sem-LSH)

The general working procedure and theoretical foundation of LSH was briefly explained in chapter 2, section 2.3.1. The procedure for indexing short essays in the short essay data set $S_e$ using LSH can be done as follows [83, 106]:

1. Select $k$ hash functions $h$ randomly and uniformly from the LSH hash function family $H$ and create $L$ buckets for each hash function.
2. Create a hash table by hashing all short essay $s_e$ in the short essay data set $S_e$ into different buckets based on their hash values.

The procedure for searching similar short essays to new short essay $u_a$ is as follows:

1. When a new short essay $u_a$ arrives, use the same set of $k$ hash functions $h$ to map $U_a$ into $L$ buckets, one from each hash table.
2. Retrieve all short essays $s_e$ from the $L$ buckets, collect them into a candidate set $C$, and remove duplicate points in $C$ if any.
3. For each short essay $s_e$ in $C$ compute its distance to $u_a$ and assign the score of $s_e$ to that $u_a$ which has smaller distance from $u_a$.

Given the general working procedure of LSH, the proposed Semantic-based LSH (Sem-LSH) algorithm works in the following manner and is presented in figure 4.3. Given a set of unlabeled short essays $S_u$ and their pair-wise distances, a threshold $\theta$ to generate the similar pairs $SP$; Also given the distance threshold $\Theta$ determining near neighbours. First, learn the document vector representation using doc2vec and feed it to the intelligent k-means to cluster short essays into different clusters. Then after the human rater scores or labels the proportional amount of short essays from each cluster that will be used to
training the short essay scoring model, i.e. Semantic LSH in our case. After labeling the short essays will fall into scored short essays $S_e$ and unscored short essays $U_{a}$. Sem-LSH then creates $L$ indices as in LSH using the set of short essays $S_e$; whenever a new unscored short essay $u_{a}$ comes, Sem-LSH retrieves all points in the buckets to which $u_{a}$ is hashed as the candidate set $C$. The full working procedure of Sem-LSH is described in Algorithm 4 and figure 4.3.

4.4 Experimental Setup

4.4.1 Feature representation

We used two different neural network approaches to learn the Semitics representation of the input essays. doc2vec, which is an extension to word2vec for learning document embeddings [57]), was used to learn the feature representations as an input to the clustering algorithm. The most popular word embedding model proposed by Mikolov [67], trained using the skip-gram method by maximizing the average log probability of all words is used to learn the feature inputs to LSH.
Reducing Annotation Effort

Algorithm 4 Scoring using Sem-LSH

1: procedure INPUT:
2: a set of scored short essays $S_e$;
3: $L$ hash tables created by LSH;
4: A set $SP$ storing all similar pairs in $S_e$ whose pairwise distance are smaller than $\theta$
5: A distance threshold $\Theta$ defining near neighbors;
6: un-scored short essays $U_a$
7: for all $c \in C$ do
8: Compute the distance between $U_a$ and $c \ d(U_a, c)$
9: if $d(U_a, c) < \Theta$ then
10: Output $c$ as a near neighbor of $U_a$
11: Search in $SP$ for all essay $S'_e$ which satisfies
12: $d(U_a, S'_e) < \Theta - d(U_a, c)$
13: for all essay $S'_e$ found in $SP$ do
14: if Found then
15: Output $S'_e$ as a near neighbor of $U_a$
16: end if
17: end for
18: end if
19: end for
20: Output:
21: Sort the near neighbors using their distance from $U_a$
22: Assign the score of the nearest short essay to $U_a$
23: end procedure
4.4.2 Data Preprocessing

We tokenize the short essays using the NLTK\(^1\) tokenizer, lowercase the text, and normalize the gold-standard scores to the range of \([0, 1]\). To learn the representation of each essay, the freely available word2vec\(^2\) word embedding was used, with an embedding for 3 million words/phrases from Google News trained using the approach in [67]. During evaluating the performance, we re-scale the system-generated normalized scores to the original range of scores and measure the performance using quadratic Weighted Kappa scores described in the following subsection. Python was used to implement the proposed LSH-based AEE algorithm. The Scikit-learn\(^3\), gensim\(^4\) and the Numpy\(^5\) libraries were used.

4.4.3 Clustering

After short essays are pre-processed, and their feature vector is learned using the doc2vec model, the clusters will be created using the clustering algorithm. We used an Intelligent K-Means (IK-Means) algorithm to cluster short essays. IK-Means is a non-parametric clustering algorithm developed from the K-Means clustering algorithm [36]. It does not need prior information of cluster number. Using the IK-Means algorithm, we can obtain better cluster results which can be used by the annotator to select representatives’ short essays to be scored manually.

4.4.4 Dataset

The experiment was carried out on ten datasets provided by the Hewlett Foundation at Kaggle\(^6\) competition for automatic short essays. There are ten datasets containing student responses from grade ten students. All the datasets were rated by two human raters. Each essay is labelled with a numeric score ranging from 0.0 to 2.0/3.0, and the answer length

\(^1\)http://www.nltk.org
\(^2\)https://code.google.com/archive/p/word2vec/
\(^3\)http://scikit-learn.org/
\(^4\)https://radimrehurek.com/gensim/
\(^5\)http://www.numpy.org/
\(^6\)https://www.kaggle.com/c/asap-sas
Table 4.1 Essay sets used in the experiment and their main characteristics

<table>
<thead>
<tr>
<th>Essay Set</th>
<th>Grade Level</th>
<th>Domain</th>
<th>Score range</th>
<th>Average length in words</th>
<th>Training set size</th>
<th>Test set size</th>
<th>Total size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>Science</td>
<td>0-3</td>
<td>50</td>
<td>1672</td>
<td>558</td>
<td>2230</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>Science</td>
<td>0-3</td>
<td>50</td>
<td>1278</td>
<td>426</td>
<td>1704</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>English, arts</td>
<td>0-2</td>
<td>50</td>
<td>1891</td>
<td>631</td>
<td>2522</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>English, arts</td>
<td>0-2</td>
<td>50</td>
<td>1738</td>
<td>580</td>
<td>2318</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>Biology</td>
<td>0-3</td>
<td>60</td>
<td>1795</td>
<td>599</td>
<td>2394</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>Biology</td>
<td>0-3</td>
<td>50</td>
<td>1797</td>
<td>599</td>
<td>2396</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>English</td>
<td>0-2</td>
<td>60</td>
<td>1799</td>
<td>601</td>
<td>2400</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>English</td>
<td>0-2</td>
<td>60</td>
<td>1799</td>
<td>601</td>
<td>2400</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>Science</td>
<td>0-2</td>
<td>60</td>
<td>1798</td>
<td>600</td>
<td>2398</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>Science</td>
<td>0-2</td>
<td>60</td>
<td>1799</td>
<td>599</td>
<td>2398</td>
</tr>
</tbody>
</table>

ranges from single phrases to several sentences. The characteristics of the used datasets are shown in Table 4.1. In this work, we merge both the training and test size since we want to use a small size training set and the proportion of training and test size is determined via the experiment.

4.4.5 Evaluation Metrics

Model validation in short essay scoring systems depends on comparing the similarity between the predicted score of the model with the score given by the human raters [4]. In this scenario, the scores from human judges are considered as gold standard and function as an explicit criterion for evaluating the performance of ASEE models. We use the most widely used evaluation metric called quadratic weighted kappa (QWK). QWK is an error metric that measures the degree of agreement between the automated scores and the resolved human scores and is an analogy to the correlation coefficient. This metrics score ranges from 0 to 1. In case that there is less agreement between the graders than expected by chance, this metric may go below 0. Quadratic weighted kappa is calculated as follows. First, a weight matrix $W$ is constructed using equation 4.1.

$$W_{i,j} = \frac{(i - j)^2}{(R - 1)^2} \quad (4.1)$$
Reducing Annotation Effort

where \( i \) and \( j \) are the actual rating (assigned by a human annotator) and the predicted rating (assigned by an ASEE system), respectively, and \( R \) is the number of possible ratings.

An observed score matrix \( O \) is calculated such that \( O_{i,j} \) denotes the number of essays that receive a rating \( i \) by the human annotator and a rating \( j \) by the ASEE system. An expected score (rating) matrix \( E \) is calculated as the outer product of histogram vectors of the two (actual and predicted) ratings assuming that there is no correlation between rating scores. The matrix \( E \) is then normalized such that the sum of elements in \( E \) and the sum of elements in \( O \) are the same. Finally, given the matrices \( O \) and \( E \), the QWK score is calculated as indicated in equation 4.2.

\[
k = 1 - \frac{\sum_{i,j} W_{i,j} O_{i,j}}{\sum_{i,j} W_{i,j} E_{i,j}}
\] (4.2)

4.5 Results and discussion

The main objective of this experiment is to investigate approaches that will help raters in selecting a small proportional amount training essays to be scored by human rater manually that would be used to train an automatic scoring model and also find an approach that can work in scenarios where there is an only small amount of training data. The approach should be cheaper and with a performance comparable to scoring models created with large datasets. After tuning the parameters of the proposed Sem-LSH approach, its performance from the best value of its parameters is presented in table 4.2.

We implemented clustering-based Locality sensitive hashing (CLSH) where the short essays are first clustered to select training set and feed to LSH. The input features for the clustering are learned using term frequency inverted document frequency (TF-IDF); we also implemented an intelligent K-means clustering where the scores of each cluster centroid are propagated to other members of the same clusters. In our proposed approach the short input answers are clustered using semantic intelligent k-means, and then we select in total 40% as a training data proportionally from each cluster that will be scored
Reducing Annotation Effort

by human annotators manually, LSH is trained after their feature vector is learned using word2vec model.

Table 4.2 The resulting accuracy of the proposed LSH-based method using QWK compared with different other two baseline.

<table>
<thead>
<tr>
<th>Set</th>
<th>cluster</th>
<th>CL-LSH</th>
<th>Sem-LSH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.66</td>
<td>0.82</td>
<td>0.89</td>
</tr>
<tr>
<td>2</td>
<td>0.66</td>
<td>0.84</td>
<td>0.89</td>
</tr>
<tr>
<td>3</td>
<td>0.71</td>
<td>0.83</td>
<td>0.92</td>
</tr>
<tr>
<td>4</td>
<td>0.73</td>
<td>0.83</td>
<td>0.91</td>
</tr>
<tr>
<td>5</td>
<td>0.75</td>
<td>0.87</td>
<td>0.96</td>
</tr>
<tr>
<td>6</td>
<td>0.76</td>
<td>0.84</td>
<td>0.95</td>
</tr>
<tr>
<td>7</td>
<td>0.55</td>
<td>0.78</td>
<td>0.87</td>
</tr>
<tr>
<td>8</td>
<td>0.48</td>
<td>0.77</td>
<td>0.86</td>
</tr>
<tr>
<td>9</td>
<td>0.68</td>
<td>0.80</td>
<td>0.87</td>
</tr>
<tr>
<td>10</td>
<td>0.69</td>
<td>0.82</td>
<td>0.89</td>
</tr>
</tbody>
</table>

The results presented in table 4.2 for both CL-LSH and Sem-LSH are only the best ones which are obtained by after tuning window size of a range $[2 - 8]$, with permutation number of $[20 - 120]$ with different size of training data ranging from $[20\%-40\%]$. We experimented with different value combinations of the parameters. After comparing the accuracy at different values of the Sem-LSH parameters, the Sem-LSH approach performed well in terms of classification accuracy where the number of permutation is equal to 90; window size equal to 5 and similarity threshold equal to $75\%$ by using only $40\%$ of annotated training essay set. An average weighted kappa score of $90.2$ was obtained. The performance concerning different parameter values is presented in the following subsections.

Accuracy with Varying Size of training set

During training Sem-LSH, we varied the size of the training set in the range between $20\%$ and $40\%$. As the size of the training set increases, the performance of Sem-LSH predicting the score of unscored short essays was increasing as well in a very high rate. Nevertheless, the accuracy of prediction starts to show stable results when the training set was between $30\%$ and $40\%$. The accuracy reported in table 4.2 was obtained using the training set size.
Reducing Annotation Effort

of 40%. We did not experiment with training size greater than 40% since the objective is to investigate the possibility of training ASEE with small amount of labelled short essays.

Varying window Size and Number of Permutations

The accuracy of Sem-LSH was increasing with the increase of window size from 2 to 5 and started to decrease when it further increase from 5 and above. The best accuracy was obtained with the window size equal to five. We did not report the accuracy of Sem-LSH for window size less than five and greater than five as the accuracy was not better.

From our experimental results, we can deduce that ASEE engines can be trained with small training short essay sets, and they can perform well like those engines trained with relatively high training essay sets. This way, we can decrease the annotation efforts required to create a training essay set and also biases during annotation due to human annotation.

4.6 Summary

In this chapter, we presented an approach that minimizes the efforts involved in creating annotated instances when training supervised automatic short essay scoring. Rather than letting a teacher annotate instances at random, we argue that by carefully selecting the instances to be annotated on with the help of an unsupervised machine learning approach, a comparable model can be trained at a much lower cost. We address the problem by using semantic intelligent k-means clustering of the short essays and let the teacher proportionally annotate only the cluster centroids and other members in each cluster until the required amount is annotated. The remaining majority of the short essays in each cluster are scored using the semantic LSH trained on these annotated short essays. The performance of the Sem-LSH based ASEE system was evaluated and compared with two other baselines. The best accuracy of 90.2% was achieved by using only 40% of short training responses selected with semantic intelligent k-means and manually annotated by the teacher. Therefore, ASEE systems can be trained by selecting and annotating a small set of representative training short essay sets selected using the semantic clustering
Reducing Annotation Effort

approach and evaluated using Sem-LSH methods with an accuracy comparable to that of trained using a large amount of training short essays.
Chapter 5

Semantic-Based Feedback Recommendation

5.1 Motivation

Existing commercial and open source e-Learning platforms offer functionalities to help teachers and support learners. Although e-Learning has come a long way, some of its aspects are still in their early stages. One such aspect is e-Testing, offered by most of the platforms. However, these e-Testing authoring tools have limited functionalities.

Using e-Testing modules, e-Learning platforms can provide both subjective and objective exams. We denote an exam “objective” if it has objective evaluation criteria (e.g. computing a mathematical equation or choosing from several options which might be correct or not). On the contrary, we call an exam subjective if the correctness of its solution depends on subjective preferences of the evaluator (e.g. an essay about a specific topic where not only the information contained are evaluated but also it is writing style or some other characteristics). However, providing appropriate feedback (score and textual comment) timely, especially for the subjective type of exams, is a challenging task for the teacher and is not yet addressed well in most of the e-Testing modules.

Moreover, for the solutions students submit through e-Testing systems, providing automated feedback suggesting a possible solution are also not well addressed in the
literature. On the other hand, e-Testing systems should also provide additional features that help teachers to highlight parts of a text (student solution) and provide a comment related to this part. Adding these features into current e-Testing systems might not be challenging, but making them, in some sense, intelligent is an open research issue. Rather than letting the assessor give feedback to all student solution in a pen and pencil way, which is a high burden to a human and also time-consuming, we need to develop algorithms that will choose some representative student solutions for which the assessor would give feedback (in the form of textual comments). The system will automatically recommend feedback to other student solutions based on their similarity to the chosen representative solutions.

The issue related to essay scoring can be addressed by using essay evaluation systems which are used to automatically evaluate and score essay exam solutions [88, 87]. Many evaluative studies have reported relatively high levels of correspondence between the scores produced by ASEE systems and those produced by human markers [71, 30, 88]. However, despite these positive results and the potential benefits of ASEE technology, ASEE systems are yet to be widely accepted by professional educators [102, 50]. A possible reason for this is the lack of transparency of these systems [27] in the results produced, leading to assessors’ low confidence in the validity of ASEE systems.

Issues on ASEE systems (acceptance) and the issue of providing textual comments on student solution can be addressed using the following manners: One way is to allow the assessors to interact with ASEE systems by crosschecking the machine score and allowing to modify this score. The second solution is to incorporate a feedback recommendation system that allows the assessor to select the part of a student solution, give comments on this part and the feedback recommendation system will recommend the most likely comments for other student solutions provided. Therefore, in this work, we proposed a semantic-based feedback recommendation approach for ASEE systems that will allow the assessors to interact with systems, allow them to give feedback and the proposed approach will provide a recommendation to other similar essays based on their similarity to the solution which has been evaluated by the assessor.
Semantic-Based Feedback Recommendation

The rest of this chapter is organized as follows: Section 5.2 provides an overview of the current works and approaches. In Section 5.3, the proposed semantic-based feedback recommendation approach is introduced. Experiments and results are described in Section 5.4. Section 5.5 presents the summary of the chapter.

5.2 Related Work

According to AL-Smadi and Gütl [1], the reasons for using e-Testing instead of pen-and-pencil tests are both practical and pedagogical. The practical ones are given by the increase in students numbers and, implicitly, of assessors’ quantity of work. The e-Testing is meant to resolve the problem of evaluation of a large number of students in a short period. The pedagogical reasons come from the need for systems which evaluate students’ knowledge correctly and efficiently. In the past, the purpose of e-Testing systems was to shorten the time spent by the teachers for the evaluation process, but now, e-Testing systems have new challenges to overcome: the efficient management of questions, the building of intelligent tests, and providing timely feedback to learners in the form of textual comments.

The research on automatic essay exam evaluation and scoring essay exams on the e-learning platform is ongoing for more than a decade where Machine Learning (ML) and Natural Language Processing (NLP) techniques were used for evaluating essay exams. The history of developing AEE systems started in 1966 by[71] and followed by other research works like E-Rater [5, 4], Intelligent Essay Assessor (IEA) [30], IntelliMetric [81], Pairwise [88] and others. These systems automatically score essays, but assessors are not allowed to interact and give feedback in the form of textual comments on each student’s solutions.

e-Learning systems use different recommendation techniques in order to suggest online learning activities to learners, based on their preferences, knowledge and the browsing history of other learners with similar characteristics. Recommender systems assist the natural process of relying on friends, classmates, lecturers, and other sources of making the choices for learning [62]. Most of the works on recommendation techniques in education
are focused on recommending learning materials or learning activities to the learners [34, 53]. On the other hand, educational data mining has considered supporting universities, teachers and learners by helping the learners improve their performance, to know how the learners learn and how they adapt to new problems [93].

There were also attempts to develop recommender systems that provide feedback to the students’ essay submissions in the form of textual comments. Gibbs and Simpson [31] describe several conditions under which feedback has a positive influence on learning. Feedback should be (i) sufficient in frequency and detail (ii) focused on students’ performance, on their learning and on the actions under students’ control rather than on the students themselves and/or on personal characteristics, (iii) timely, in that it is received by students while it still matters and in time for application or for asking further assistance, (iv) appropriate to the aim of the assignment and its criteria, (v) appropriate in relation to students’ conception of learning, their knowledge and of the discourse of the discipline (vi) attended to and (vii) acted upon.

The Altered Vista (AV) system, proposed by Recker and Walker [97], uses a database in which learner evaluations of learning resources are stored. It allows learners to browse the reviews of others and get personalized learning resource recommendations from the system. AV does not support learners directly by giving them feedback on their work. Instead, it provides indirect learning support by recommending suitable learning tools.

Another similar web-based application called PeerGrader (PG) was also introduced by [32]. PG helps learners’ to improve their skills by reviewing and evaluating solutions of their fellow learners’ blindly. In PG, each learner will get a task list and can choose a task by the PG. Learners submit their solutions and other learners can read these solutions and provide textual comments. Learners can modify solutions based on the comments received and re-submit an updated version of their solutions again to the system where other learners can review it again. After learners submitted their final updated version, the PG calculates grades for these solutions. The evaluation of a single learner answer is very time-consuming because of the complexity of the reviewing process and the textual comments. This may cause learner dropouts and deadline problems [72].
The Scaffolded Writing and Rewriting in the Discipline (SWoRD) system was introduced by [18] to address the problem of writing homeworks in the form of a long text which cannot be reviewed in detail by a teacher for time reasons. SWoRD relies on peer reviews. Students who conduct peer reviews, read possible task solutions that were provided by other students and evaluate them. Based on peer reviews, the system provides feedback to learners in the form of recommendations.

Sourcer’s Apprentice Intelligent Feedback system (SAIF) [14] used Latent Semantic Analysis (LSA) and Regular Expression Pattern to provide feedback for students to write essays. The system can be used to detect plagiarism, uncited quotations, lack of citations, and limited content integration problems. The most relevant work to the present study is Glosser, which is an automatic writing feedback system that provides academic essay writing support for college students [16, 65]. It uses text mining algorithms to analyze various features of texts, based on which feedback is provided to student writers. Glosser provides feedback on some aspects of the writing, such as flow, topics, and topic map visualization. The feedback is given in the form of generic trigger questions and document features that relate to each set of questions.

The works by Andersen et al. [2] is another system that provides model-driven sentence-level feedback focused on mechanics and usage. Their models require sentence-level annotations of grammatical errors. The authors assign pseudo scores to the sentences based on the whole essay score and the count of mistakes within the sentence. Once the model is trained directly on these sentence-level scores, it will be used to predict the scores of a new sentence and provide feedback. More similar work to Andersen et al. [2] is the one introduced by Adamson et al. [104] that provides rubric-specific, sentence-level feedback to students to supplement teacher guidance with sentence-level annotations.

Writing Pal is an intelligent tutoring system for scaffolds writing and feedback within learning tasks by Roscoe et al. [78]. It is automated holistic feedback targets student writing strategies using highly engineered essay features and a series of algorithms for each feedback type, separate from the scoring model itself.
The works by Liu et al. [61] focused on more aspects of the writing, such as Grammar, Spelling, Sentence Diversity, Supporting Ideas and Organization, since these aspects were frequently addressed in the teachers’ feedback in the context of Chinese English as a second language learners’ writing based on our empirical study findings. They used Coh-Metrix to extract features to build the feedback classification model using a supervised machine learning approach to classify the quality of essays regarding to each writing aspect.

Some of the previous works provide formative feedbacks based on peer reviews that are time-consuming for the learner to wait and to get the feedback and also the feedback might not even be useful because their fellow learners’ might do the review blindly. The remaining works in the literature provide formative feedback on students writing focused aspects of the writing, such as Grammar, Spelling, Sentence Diversity, Supporting Ideas, and Organization. Therefore, we proposed a semantic-based formative feedback recommendation approach for ASEE systems. Our approach allows assessors to provide formative feedback in the context of the question in the form of textual comments on selected student essay solutions and make recommendations to other similar essay solutions based on evaluated solutions by the assessor.

5.3 Feedback Tag Recommendation

As discussed in sections 5.1 and 5.2, in we will address an open research problem called semantic-based feedback recommendation. Our e-Testing system [87] has different features that help both the assessors and learners( please check appendix A for more ). It allows assessors to create their own course, add students they want to assess, create both subjective and objective exams for the course and it also allows the assessors to give feedback in the form of textual comments on selected (and highlighted) parts of students submissions. The system allows learners to register and get their own username and password, send a request to register for course, Submit their solution for the registered exams and view their score and the assessor’s feedback.
The feedback system is the same as a manual way of giving textual feedback to the learners (formative feedback). To make the feedback feature of the system more useful and supportive towards the assessors, we introduce a semantic-based feedback recommendation approach which works in the following manner: First, the assessor gives textual comments on some essays by selecting a phrase, a sentence or a paragraph. On each students essay, the assessor can give as many comments as he/she wants. Then, the system will find similar essays of other students, check whether the highlighted text (or a very similar text) is present in those essays and give recommendation for comments. To address the issue, we have proposed and implemented a semantic-based feedback tag recommendation approach. We also implemented two baseline solutions, introduced in algorithms 5 and 6, for experimental purposes.

The first baseline is a simple pattern matching approach, introduced in the Algorithm 5, that has the following data on its input:

- a student essay \( e \), a text, for which the feedbacks are going to be recommended,
- the set of all essays \( E \) submitted by all the students not evaluated so far by the teacher, such that each essay correspond to one student (i.e. \( |E| \) is the number of students),
- the set \( F = \{(h, c) \mid h \text{ is a text}, c \text{ is the teacher’s comment to } h \} \) of teacher’s feedbacks present in already evaluated essays, i.e. comments \( c \) on some (parts of) texts \( h \) from students’ essays evaluated by the teacher.

The algorithm looks up in a so far not evaluated essay \( e \) if its parts are identical to already commented parts of other essays on the same topic.

The second baseline, introduced in the Algorithm 6, is based on semantic and lexical similarity with the same data on input as in the case of Algorithm 5 and a similarity threshold \( \theta \) what is a hyper-parameter of the algorithm and has to be set up by the user (might require a certain domain knowledge). Equations 5.1 and 5.2 are used in algorithm 6 to compute the similarity.
Algorithm 5 General Rule based algorithm

1: procedure TAG RECOMMEND(e, E, F)
2:     for all e ∈ E do
3:         e.sentences = tokenize(e)
4:     for all (h, c) ∈ F do
5:         for all s ∈ e.sentences do
6:             if h ⊆ s then
7:                 F ← F ∪ (s, c)
8:         end if
9:     end for
10: end for
11: end procedure

Algorithm 6 General Similarity based algorithm

1: procedure TAG RECOMMEND(e, E, F, θ)
2:     for all e ∈ E do
3:         e.sentences = tokenize(e)
4:     for all (h, c) ∈ F do
5:         for all s ∈ e.sentences do
6:             if similarity(h, s) ≥ θ then
7:                 F ← F ∪ (s, c)
8:         end if
9:     end for
10: end for
11: end procedure
5.3.1 Lexical Similarity

The lexical similarity between two text documents can be easily computed using pairwise similarity metrics. One of the most widely used lexical similarity measure in text document similarity is cosine similarity. Given the set of all essays $E$ submitted by all the students not evaluated so far by the teacher, sentences $s_1, s_2, \ldots, s_n$ of a student essay $e \in E$ and the set $F = \{(h, c) \mid h \text{ is a highlighted text, } c \text{ is the teacher’s comment to } h\}$ of teacher’s feedbacks present in already evaluated essays, i.e. comments $c$ on some (parts of) texts $h$ from students’ essays evaluated by the teacher, the cosine similarity between each $s_j$ (where $1 \leq j \leq n$) and a given $h$ is defined as follows:

$$cosine\_sim(s_j, h) = \frac{x_{s_j} \cdot x_h}{\|x_{s_j}\| \|x_h\|} \quad (5.1)$$

where $x_{s_j}$ is a vector representation of sentence $s_j$ of the essay $e$ and $x_h$ is a vector representation of a highlighted text $h$.

5.3.2 Relaxed Word Movers based similarity

To compute the semantic similarity between sentences in essays and highlighted text, we use the word mover distance [56] and we redefine it as a relaxed word mover similarity(RWMS) using cosine similarity for fast similarity computation. A much faster results can be obtained by relaxing the WMD optimization problem defined in 3.1 and removing one of the two constraints.

The RMWS utilizes the property of word2vec embeddings [67]. Therefore, the similarity between two sentences $s_1$ and $s_2$ is the maximum cumulative similarity that word vectors from sentence $s_1$ travels to match exactly to word vectors of sentence $s_2$.

In this regard, in order to compute the semantic similarity using the RWMS between sentences $s_1, s_2, \ldots, s_n$ contained in an essay $e \in E$ not evaluated so far by the teacher and $h$ a highlighted text from the set $F$ of teacher’s feedbacks present in already evaluated essays, defined above, $s_1, s_2, \ldots, s_n$ will be mapped to $h$ using a word embedding model. Let $s_j$ and $h$ be nBOW representations of $s_j$ and $h$, respectively, where $1 \leq j \leq n$. Let $T \in \mathbb{R}^{m \times m}$
be a flow matrix, where $T_{kl} \geq 0$ denotes how much the word $w_k$ in $s_j$ has to “travel” to the word $w_l$ in $h$, and $m$ is the number of unique words appearing in $s_j$ and $h$. To transform $s_j$ to $h$ entirely, we ensure that the complete flow from the word $w_k$ to the word $w_l$ equals to some value $d_k$. The relaxed word movers similarity is defined as follows using cosine similarity measure:

$$ \max_{T \geq 0} \sum_{k,l=1}^{m} T_{kl} \cos_sim(w_l, w_k) \quad (5.2) $$

subject to

$$ \sum_{k=1}^{m} T_{kl} = d_k, \forall i \in \{1, \ldots, n\} $$

### 5.4 Experimental study

#### 5.4.1 Dataset selection and preparation

In order to do the experiment, a data set which has highlighted texts and comments on the students’ essay is mandatory. However, we could not find such a data set, probably because there was no attempt to address such an issue so far. Therefore, for this experiment, we have simulated the process of teacher evaluating (i.e. highlighting and commenting) essays. We used the benchmark data provided by the Hewlett Foundation at Kaggle\(^1\) for an AEE competition. There are 10 data sets, corresponding to 10 different topics, in the benchmark containing student essays. Two human raters rated all the data sets.

From these data, we randomly selected 2000 essays belonging to four different datasets, i.e. essay topics, for this experiment. Then, we randomly selected 6 different essays in which we manually highlighted a randomly selected sentence and generated a comment for this highlighted sentence. This simulates a teacher in evaluating 6 essays on different topics such that in each essay he/she comments one selected (highlighted) sentence.

Then, we have searched the similar sentences in all the 2000 essays to those 6 sentences which were highlighted manually. We did it by a simple syntactic similarity search such that sentences which are 100% similar to those 6 sentences were highlighted and manually

\(^1\)https://www.kaggle.com/c/asap-sas
assessed by a human. The resulting set of 2000 essays each containing at least one highlighted text and a related comment serves as the experimental dataset for the proposed approaches. It is important to note that in the resulting dataset, there are 6 different feedbacks corresponding to 6 different texts. We will denote these 6 sentences and related comments as “feedback one” (shortened as F1), “feedback two” (shortened as F2), . . . , “feedback six” (shortened as F6), respectively.

5.4.2 Data Preprocessing

In preprocessing an essay, the following tasks were performed: tokenization; removing punctuation marks, determiners, and prepositions; a transformation to lower-case; stopword removal and word stemming. In the stop word removal step, the words that are in the stop word list [39] were removed. After removing the stopwords, the words have been stemmed to their roots.

For essay evaluation, the freely available word2vec word embedding, which has an embedding for 3 million words/phrases from Google News trained using the approach in[67], was used as a word embedding model in the implementation of the proposed approach.

Python was used to implement the proposed semantic-based feedback recommendation algorithms and other baseline algorithms discussed above. As the Relaxed Word Mover’s Similarity (RWMS) algorithm is dependent on a word embedding, we used the freely-available Google News word2vec model. Also, Scikit-learn and Numpy Python libraries were used.

The performance of the proposed semantic-based feedback recommendation system is compared to the rule-based and cosine similarity [28, 105] approaches described in the Algorithms 5 and 6.

2https://code.google.com/archive/p/word2vec/
3http://scikit-learn.org/
4http://www.numpy.org/
5.4.3 Evaluation Metrics

To assess the successful decision-making capacity of the feedback recommendations algorithm and to evaluate the recommendation quality, we used classification accuracy metrics. Precision and recall are the most popular metrics used for evaluating information retrieval systems and recommender systems [80].

They are used to measure the amount of correct and incorrect classifications as relevant or irrelevant feedback that is made by the recommender system and are therefore useful for learning tasks such as finding good and relevant feedbacks in e-Testing systems.

**Precision:** (also called confidence in data mining) is a measure of exactness or fidelity and is calculated as the ratio of recommended feedbacks that are relevant to the total number of recommended feedbacks. This is the probability that recommended feedback corresponds to the learner’s solution. A precision score of 100% would indicate that every recommendation retrieved was relevant [8].

If $AF = \{(h_a, c_a)\}$ denotes the set of actual feedback tags (i.e. highlighted texts and related comments) by the assessor and $RF = \{(h_r, c_r)\}$ denotes the set of recommended or predicted feedback tags by the recommender system, then Precision can be defined as in the Equation 5.3:

$$
Precision = \frac{|AF \cap RF|}{|RF|} \quad (5.3)
$$

**Recall:** (also called sensitivity in psychology) is a measure of completeness. Recall score of 100% would indicate that all relevant recommendations were retrieved. Recall is calculated as the ratio of recommended feedbacks that are relevant to the total number of relevant feedbacks [8, 80]. This is the probability that a relevant feedback is recommended and is defined as follows in equation 5.4:

$$
Recall = \frac{|AF \cap RF|}{|AF|} \quad (5.4)
$$

**F1-measure:** Since both Recall and Precision are important in evaluating the performance of a system which generates relevant recommendations, they can be combined to
Table 5.1 Performance of feedback tag based recommendation using Precision, Recall and F1-Score for dataset

<table>
<thead>
<tr>
<th>Methods</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semantic Based (RWMS)</td>
<td>0.925207</td>
<td>0.918042</td>
<td>0.914620</td>
</tr>
<tr>
<td>Lexical Based</td>
<td>0.742056</td>
<td>0.691038</td>
<td>0.635401</td>
</tr>
<tr>
<td>Rule Based</td>
<td>0.746979</td>
<td>0.688090</td>
<td>0.631034</td>
</tr>
</tbody>
</table>

get a single metric, the F1-measure, which is a weighted combination of Precision and Recall [8], and is defined as follows in equation 5.5.

\[
F1\text{-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]  

5.4.4 Experimental results and Discussions

Results in figure 5.1 show the comparison between the actual feedbacks and predicted (recommended) feedbacks. RWMS has an accuracy of 0.92, 0.97, 0.95, 0.99, 0.96 and 0.61 in correctly recommending F1, F2, F3, F4, F5 and F6, respectively.

The results of lexical based feedback recommendation algorithms have an accuracy of 0.12, 0.18, 0.44, 0.99, 0.98 and 0.58 in correctly recommending F1, F2, F3, F4, F5, F6, respectively, according to figure 5.2. According to figure 5.3 the rule-based feedback recommendation algorithms have an accuracy of 0.12, 0.16, 0.44, 0.99, 0.97 and 0.57 in correctly recommending the related feedbacks F1, F2, . . . , F6, respectively. Therefore, according to the the results of 5.1, 5.2 and 5.3 the proposed semantic based feedback tag recommendation algorithm has outperformed the baseline algorithms.

Table 5.1 shows the performance of the algorithms using Precision, Recall and F1-measure. Using the semantic-based feedback recommendation, 92% of retrieved recommendation is relevant, and 91% of relevant recommendation is retrieved while 74% of retrieved recommendation is relevant; and 69% and 68% of relevant recommendation is retrieved using rule-based and lexical based algorithms respectively. The results in table 5.1 also show that the semantic-based algorithm has outperformed the baseline algorithms.
Figure 5.1 Normalized Confusion matrix which shows the number of the cases from actual recommendation are correctly predicted (recommended) and how many of the cases are incorrectly predicted (recommended) by the RWMS algorithm.

Figure 5.2 Normalized Confusion matrix which the number of the cases from actual recommendation are correctly predicted (recommended) and how many of the cases are incorrectly predicted (recommended) by the Lexical based algorithm.
In general, the results show that rule-based recommender algorithm and lexical-based recommender algorithm can work well in cases where there is a word by word mapping between a sentence $s$, from the set of all essays $E$ submitted by all the students but not evaluated so far by the teacher, and a highlighted text $h$ from the set $F = \{(h,c)\}$ of teacher’s feedbacks present in already evaluated essays, i.e. comments $c$ on some (parts of) texts $h$ from students’ essays evaluated by the teacher. Semantic similarity-based recommender algorithms can work by understanding the meaning behind them. That is the main reason for semantic similarity-based recommender algorithms to have a total accuracy of 91.8% while lexical-based and rule-based recommender algorithms have an accuracy of 69.1% and 68.8%, respectively.
5.5 Summary

Today, many e-Learning platforms offer authoring tools for e-Testing. These authoring tools allow assessors to create essay exams, and the scoring will be done either automatically or manually. In most of the cases, the authoring tools do not have an option for the teacher to give feedback in the form of textual comments. To address such issues, we have successfully hosted the e-Testing system where the assessor can give feedback in the form of textual comments. To make this feature more useful and helpful, a semantic-based feedback recommendation approach was proposed and implemented. The proposed algorithm uses ground truth feedbacks from the assessor to give feedback recommendations to other similar student essay solutions. To compute the semantic similarity between sentences, we used a relaxed word movers similarity distance that computes semantic similarity based on neural word embedding. The experiment was carried out in 2000 randomly selected essays from 10 different datasets which were provided by Kaggle for automatic essay evaluation, and we simulate the role assessors to obtain textual feedbacks. The performance of the proposed approach was evaluated and compared to other state-of-the-art algorithms. According to the experimental results, the proposed approach has outperformed the baseline algorithms.
Chapter 6

Conclusion and Recommendation

6.1 Contributions

In this dissertation, we have addressed several problems related to automated essay scoring systems and have proposed solutions to these problems. In this section, we briefly summarize the main contributions to scientific engineering contributions listed in Section 1.3. With each contribution we list the sections where the topic is discussed. In addition, we also list our publications that discuss the topic. Note that the listed references were published in peer-reviewed scientific journals or presented on international conferences and were, thus, reviewed and discussed.

1. Unsupervised semantic-based short essay scoring method: We proposed an unsupervised and semantic-based essay scoring system called a layered and Pairwise short essay scoring system in the short “Pairwise” approach. The task of the pairwise approach is to predict the score for the student answer given the reference answer. Instead of using input representation based on bag-of-words, Pairwise considers both the student answer and reference as a sequence of words with rich contextual structure, and it retains maximum contextual information in its projected latent semantic representation by projecting each word in its context onto a low-dimensional continuous feature vector using skip-gram models. Then, the layered pair-wise method computes the semantic similarity between the feature vector representation of the
Conclusion and Recommendation

reference solution and a student answer using Relaxed word mover’s similarity (RWMS) method. We described the details in section 3.3, 3.3.1 and also in paper [88] and [89].

2. **A new and semantic way of reducing annotation effort in short essay scoring:**
We proposed an efficient way of selecting a small-sized training set to be annotated by human rater that will be used to train the scoring engine using semantic intelligent k-means and semantic Locality Sensitive Hashing (Sem-LSH) techniques. Specifically, we investigate the question of what proportion of the short essays should the teacher score manually before the ASEE engine is used on new short essays without labels (scores) and how to select those responses proportionally. We introduced a new way of addressing the issue of reducing the workload of human graders in creating training sets for supervised ASEE. We described the details in section 4.3.4 and in paper [92]

3. **A new semi-supervised and semantic-based feedback recommendation for short essay evaluation:** We introduced a semantic-based feedback recommendation approach for automatic short essay grading that will allow the assessors to interact with systems, allow them to give feedback and the proposed approach will provide a recommendation to other similar essays based on their similarity to the solution which has been evaluated by the assessor. This work, according to our knowledge, is the first and a pioneer work in providing formative feedback to short answers. We described the details in section 5.3 and in paper [90].

### 6.2 Future Research Directions

The work in this dissertation is primarily applied and practical in some aspect in nature: We conducted studies on automatic scoring of short essay using publicly available and pre-scored data sets and assessed semantic short essay scoring in three different ways: a) in terms of scoring short essays semantically using unsupervised approaches; b) reduction of human effort in scoring short essay and the performance which can be achieved using
Conclusion and Recommendation

small scored essay set; and c) the possibility of providing formative feedback using recommendation systems.

One of the future directions will be to take these findings to the classroom and evaluate the proposed methods in a real-life setting by integrating with web based platforms which will help us to learn the advantages of automatic scoring for a teacher in an exam situation. Doing this will help the researches to learn the real demands of the teachers in developing a more useful and practical systems that will address the gas in the current scoring systems.

The second future direction of the research is to investigate and incorporate features that will increase the usability of short automatic essay evaluation. In other words, every student answer has to pass through a proper validation mechanism before computing the score because every well-written essay that does not address the question topic and also answers copied from other students may receive a good score from a scoring system. Therefore, investigating possible approaches in identifying such inputs will be one of the future research directions in the domain to increase the usability of automatic short essay scoring systems.

The third and final future direction of research will be the task of providing formative feedback to the student’s answer. We already introduced semantic feedback recommender systems that provide formative feedback to student answers. Our work, as far as we are aware of it, is the first and pioneer in providing formative feedback to short essay answers but provides a maximum of three feedbacks using manually annotated data. This means, more investigation and research are required in providing formative feedback which will give students more information in the nature and source of the error with explanations and also ways to improve the answers as well.
BIBLIOGRAPHY

Bibliography


98


Summary

In our work, we analyzed several problems related to automated short essay Evaluation and proposed new approaches to address these problems. First, we proposed an unsupervised and semantic-based essay scoring approach called Pairwise. The task of the pairwise approach is to predict the score for the student answer given the reference answer by considering both the student answer and reference as a sequence of words with rich contextual structure, and it retains maximum contextual information in its projected latent semantic representation by projecting each word in its context onto a low-dimensional continuous feature vector using skip-gram models. Then, the pair-wise computes the pair semantic similarity using Relaxed Word Mover’s Similarity method.

Second, we proposed an effective way of selecting a small-sized training set to be annotated by human rater that will be used to train the scoring engine using semantic intelligent $k$-means and semantic Locality Sensitive Hashing (Sem-LSH) techniques. Specifically, we investigate the question of what proportion of the short essays should the teacher score manually before the ASEE engine is used on new short essays without labels (scores) and how to select those responses proportionally. We introduced a new way of addressing the issue of reducing the workload of human graders in creating training sets for supervised ASEE.

Finally, we introduced a semantic-based feedback recommendation approach for automatic short essay grading that will allow the assessors to interact with systems, allow them to give feedback and the proposed approach will provide a recommendation to other similar essays based on their similarity to the solution which has been evaluated by the assessor. This work, according to our knowledge, is the first and pioneer work in providing formative feedback to short answers.
Appendix A

The E-test Application

A.1 Motivation

The most common and traditional way of evaluating and scoring exams is done using the pen and pencil system. Paper-based examinations and pen-and-pencil evaluation are prone to different kinds of problems such as inconsistency among markers because of fatigue, loss of concentration arising from boredom and neatness of student handwriting, variables in the markers’ background (level of professional experience, prior knowledge in marking) [59], and the time it takes to complete the marking process. These factors are especially present in case of evaluating solutions for so-called subjective types of exams where the evaluation is not based on some objective criteria but the teacher’s subjective opinion (e.g. the writing style of an essay is “weak”). In the opposite case we will call objective types of exams where the correctness of the solution can be evaluated based on objective measures (e.g. the solution for a math problem is “correct”).

Various on-line exam evaluation systems have been developed, however, these are mostly focusing on objective types of exams and achieve good performance [52] [108]. For subjective types of exams, Automatic Short Essay Evaluation (ASEE) systems have been introduced [69][4]. ASEE explores a new way of assessment that has (i) the capacity to deliver consistent and reliable scoring of exams, increased objectivity and efficiency in the
scoring operation, and, the reduction of time and financial costs associated with scoring. The main focus of this paper is on e-Testing systems with particular focus on ASEE.

After investigating the current ASEE systems and the state-of-the-art in e-Testing systems, we have designed and implemented an enhanced web-based e-Testing system. It comes with a new feature that helps both the teachers and students in the process of examination and scoring. It allows assessors to create their course, add students they want to assess, create both subjective and objective exams for this course and it also allows the assessors to give feedback in the form of textual comments on selected parts of student’s submissions. It also scores the exams automatically utilizing semantic text analytics techniques. The system allows learners to register and get their username and password, send a request to register for course, submit their solution for the registered exams and view their score and the assessor’s feedback in the form of textual comments.

A.2 Related Works

ASEE often called automatic exam scoring, is an interesting development domain that has been ongoing since the 1960s up to today [71]. ASEE systems are distinguished from each other primarily the way they evaluate essays such that either by style or by content or both. Another distinction criterion is the approach adopted for assessing style and/or content. The most common approaches found in the literature used for scoring essays are LDA, LSA, NLP, ML and Artificial Neural Network (ANN).

ASEE systems that focused on statistical approaches capture only the structural similarity of texts. The following systems, based on LSA, did more than a simple analysis of co-occurring terms. They introduced two new approaches to the problem: a comparison based on a corpus and an algebraic technique that allowed identifying similarities between two texts with different words [95].

The latest systems are based on NLP, ML and ANN techniques and can do intelligent analyzes that capture the semantic meaning of an essay. As mentioned above, the distinction is made between grading essays based on content and those based on style. AEE systems
that evaluate and score primarily based on style are the Project Essay Grade (PEG) [71] using linguistic features via proxies. Systems utilizing content are Intelligent Essay Assessor (IEA) [30] using LSA, Educational Testing Service (ETS) [4] and Conceptual Rater (C-Rater) [5] using NLP, and, Pair-wise [69] based on ANN.

Using the current AEE systems, evaluating and scoring essays can be done automatically but the assessor cannot interact with and give feedback to student answers in the form of textual comments. The students also cannot learn from their mistakes as AEE systems do not have such a feature so far. To support both scoring and the provided feedback in the form of textual feedback, we designed and implemented web-based intelligent exam management and evaluation system.

A.3 System Details

In this section, we explain the modules and sub-modules of our system in more detail. First, we list and explain the main modules and sub-modules of our system with the help of screenshots. Next, we will explain the work process of the system using the flow chart. Lastly, we present how the system automatically scores each solution semantically.

A.3.1 System Modules

The System Management Module (Maintenance) as shown in figure A.1 includes the User information, Audit logs, User management, and System security sub-modules. Using User Management, the system administrators can create and set basic information for teachers and students. Using User management and Authentication, the administrator can set the identities of the teachers and students, maintain user information, grant access rights to the different modules for each user, backup and restore the system information. The Audit logs module is used to trace the activities and interaction with the system of each user. If something went wrong, the audit logs will be used to restore the system into a normal state.
The Teacher module, shown in figure A.2, allows teachers to maintain their profile, change their log-in credentials, to add and maintain their courses, to approve the requests from students to register for the course(s) and upload course materials. The Student module, shown in figure A.3, allows students to register, obtain and maintain their log-in credentials, search and send a request for course registration to the teacher, submit solutions to exams and view their obtained score and feedback.

The Online Exam Management module allows teachers to create private and public, subjective and objective types of exams, as shown in figure A.4. Teachers can also add students to their exams, dispense the exam online and set the starting and ending time of the exam. Once the exams are released by the teacher, the students can log-in to the system and can submit their solution of the exam online within the given time-stamp, as shown in figure A.5. The students receive a machine score for essay exams automatically but it is not the final score. After the students submitted their solution, the teacher can look into

---

1. The developed system is not intended to serve as a learning management system (e.g. Moodle) but, rather, as an e-Testing framework. However, we have implemented this ability to upload course materials for teachers not using any other system.
The E-test Application

The student’s solution, give textual comments on essays, modify the score given by the machine and submit the final score.

The Feedback module is designed to be used by the teachers to give textual feedback to essay exam solutions. Using this module, the teacher can open each student’s essay, select and highlight phrases, sentences or paragraphs and write textual comments to these highlighted parts, as depicted in figure A.6. It allows the teachers to give an unlimited number of textual comments on a single essay solution. They can also modify the machine score based on their reviews. The students can also see the textual feedbacks given by their teacher, as depicted in figure A.7.
A.3.2 System Work Process

The work process of the main modules of our online essay exam management and evaluation system is depicted in figure A.8. It shows how the system works as well as the process of how the two main users, teacher and student, use the system.

A.3.3 System Implementation

The system was developed using web application development tools like Extended JavaScript (EXTJS), Hypertext Markup Language (HTML5.0), Cascading Style Sheet (CSS), Hypertext Pre-processor (PHP) and MySQL.

A.4 Automatic scoring Process

The automatic short essay evaluation and scoring process of the system is based on the approach introduced in chapter 3 section 3.3. Eight teachers, who were, teaching introduction to computer science courses for both majoring in computer science and non-
The E-test Application

computer science students have participated during the pilot test. The average kappa score of 0.79 was achieved during the pilot evaluation.

A.5 System Evaluation

We asked a series of questions to some of the instructors of the Wollo University, Faculty of Informatics, who have been using our system in their teaching process. We have asked them to evaluate our system. The results of the survey are reported in figure A.9, figure A.10, figure A.11, figure A.12 and table A.1. In all of the figures, we ignored zero-response answer choices to be included in the graph.

The majority of the users replied well and, particularly, they expressed that our system meets their needs. The majority of the users also said that the system is useful and reliable. The details for each survey are reported in the graphs and tables below alongside the survey questions.
Table A.1 Survey results collected from the three questions which have the same answer choices (Q1-Does the system help you grade more fairly?, Q2-Does the system save you time in grading? and Q3-Does the system make the evaluation of essay questions more easily?).

<table>
<thead>
<tr>
<th>Answer choices</th>
<th>Q1 Responses</th>
<th>Q2 Responses</th>
<th>Q3 Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>47.67%</td>
<td>33.33%</td>
<td>33.33%</td>
</tr>
<tr>
<td>Agree</td>
<td>33.33%</td>
<td>45.00%</td>
<td>40.00%</td>
</tr>
<tr>
<td>Neutral</td>
<td>13.33%</td>
<td>15.00%</td>
<td>20.00%</td>
</tr>
<tr>
<td>Disagree</td>
<td>6.67%</td>
<td>6.67%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Strongly Disagree</td>
<td>0.00%</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

### A.6 Summary

A web-based online essay exam management and evaluation system were designed and implemented as a tool to help students and teachers to create and evaluate essay exams. The system allows the teacher to give textual feedback in the form of a comment by selecting parts of the essay solution which he/she assumes is not correct. The system has five main modules which are working together: System Management module (Maintenance), Instructors module, Students module, Online Exam Management module, and the Feedback module.

The automatic evaluation and scoring of essay answers are performed using a pair-wise automatic essay evaluation method. The Pair-wise approach uses a reference answer (RA), provided by the teacher, and a student answer (SA) to compute the semantic similarity between them and give a score according to the weight of the question. In a survey, conducted to assess the feasibility and usability of the system, teachers responded that the system helps them to provide higher quality feedback in less time than using traditional paper-based grading. These preliminary, test runs are promising and show good indications for further improvement of the developed system.
EÖTVÖS LORÁND TUDOMÁNYEGYETEM

DECLARATION FORM for disclosure of a doctoral thesis

The data of the doctoral thesis:

Name of the author: Tashu Tsegaye Misikir

MTMT-identifier: 10057748

Title and subtitle of the doctoral thesis: Semantic-based Automatic Short Essay Evaluation and Recommendation System

DOI-identifier: 10.15476/ELTE.2020.119

Name of the doctoral school: Computer Science

Name of the doctoral programme: Data Science and Engineering

Name and scientific degree of the supervisor: Dr. Tomas Horvath

Workplace of the supervisor: ELTE, Budapest

II. Declarations

1. As the author of the doctoral thesis,

   a) I agree to public disclosure of my doctoral thesis after obtaining a doctoral degree in the storage of ELTE Digital Institutional Repository. I authorize Adina Kulcsár, the administrator of the ELTE Doctoral School of Informatics to upload the thesis and the abstract to ELTE Digital Institutional Repository, and I authorize the administrator to fill all the declarations that are required in this procedure.

   b) I request to defer public disclosure to the University Library and the ELTE Digital Institutional Repository until the date of announcement of the patent or protection. For details, see the attached application form;

   c) I request in case the doctoral thesis contains qualified data pertaining to national security, to disclose the doctoral thesis publicly to the University Library and the ELTE Digital Institutional Repository ensuing the lapse of the period of the qualification process.;

1 Filled by the administrator of the faculty offices.
2 The relevant part shall be underlined.
3 Submitting the doctoral thesis to the Disciplinary Doctoral Council, the patent or protection application form and the request for deferment of public disclosure shall also be attached.
4 Submitting the doctoral thesis, the notarial deed pertaining to the qualified data shall also be attached.
d) I request to defer public disclosure to the University Library and the ELTE Digital Institutional Repository, in case there is a publishing contract concluded during the doctoral procedure or up until the award of the degree. However, the bibliographical data of the work shall be accessible to the public. If the publication of the doctoral thesis will not be carried out within a year from the award of the degree subject to the publishing contract, I agree to the public disclosure of the doctoral thesis and abstract to the University Library and the ELTE Digital Institutional Repository.³

2. As the author of the doctoral thesis, I declare that
a) the doctoral thesis and abstract uploaded to the ELTE Digital Institutional Repository are entirely the result of my own intellectual work and as far as I know, I did not infringe anyone’s intellectual property rights;
b) the printed version of the doctoral thesis and the abstract are identical with the doctoral thesis files (texts and diagrams) submitted on electronic device.

3. As the author of the doctoral thesis, I agree to the inspection of the thesis and the abstract by uploading them to a plagiarism checker software.

Budapest, July 31, 2020

Signature of thesis author

³ Submitting the doctoral thesis, the publishing contract shall also be attached.