Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Jesper Trøan
Julian Nyland Skattum

Automatic Evaluation of Short Text Answers with Feedback Techniques to Enhance Student Learning Performance

Master's thesis in Informatics
Supervisor: Trond Aalberg
June 2023

NTNU
Norwegian University of
Science and Technology

Jesper Trøan
Julian Nyland Skattum

Automatic Evaluation of Short Text Answers with Feedback Techniques to Enhance Student Learning Performance

Master's thesis in Informatics
Supervisor: Trond Aalberg
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Evaluation of exams can be both challenging and time-consuming, and teachers dedicate a substantial amount of their time to this task. Consequently, there is a consensus among researchers that computer technology can be leveraged to automate this process. This thesis is a study into the field of automatic evaluation and feedback generation of student short answers. Two datasets featuring short answers written by students from a computer science course have been analyzed and utilized to explore how Information Retrieval and Text Mining techniques can be applied to automatically grade and generate feedback in the Norwegian language.

Research within the field of automatic grading has been explored for decades. However, automatic grading of unstructured text remains a challenge and minimal work is present in the Norwegian language. Hence, this thesis aims to answer what the state-of-the-art within automatic feedback generation and grading of text-based answers is to this day. Moreover, Information Retrieval and Text Mining techniques are explored to investigate how automatic grading and feedback systems can be implemented, and what quality can be achieved.

A literature study in the field of automatic grading and feedback generation is presented, as well as a summary of relevant Information Retrieval and Text Mining techniques. Two approaches for automatic grading and feedback generation have been proposed and implemented. The first compares Student Answers with Reference Answers by a similarity calculation, and the second utilizes the k-means clustering algorithm to group similar responses. Automatic grading is attempted by exploring achieved performance when applying various Information Retrieval techniques. Feedback generation is proposed as providing keywords to students representing important concepts or themes they might have missed in their responses to improve learning. Lastly, two evaluation metrics for the proposed feedback technique are proposed and discussed.

This thesis presents valuable insights into how Norwegian and multi-lingual datasets can be cleaned and processed, along with two promising approaches that may prove useful and could be utilized in a full-scale automatic grading and feedback system in the future.

Sammendrag

Evaluering av eksamener kan være både utfordrende og tidkrevende, og lærere bruker en betydelig del av tiden sin på denne oppgaven. På dette grunnlaget er det enighet blant forskere at datamaskiner kan brukes til å automatisere denne prosessen. Denne masteroppgaven er en studie av automatisk vurdering og generering av tilbakemeldinger på korte elevsvar. To datasett med korte svar skrevet av studenter i et informatikkurs har blitt analysert og brukt for å utforske hvordan informasjonsgjenfinning (Information Retrieval) og tekstdatautviklingsteknikker (Text Mining techniques) kan brukes til å automatisk vurdere og generere tilbakemeldinger på kortsvarsoppgaver på norsk.

Forskning innen automatisk vurdering har vært utforsket i flere tiår. Imidlertid forblir automatisk vurdering av ustrukturert tekst en utfordring og lite arbeid finnes på det norske språket. På dette grunnlaget er denne masteroppgaven et forsøk på å besvare hva som er det nyeste (state-of-the-art) innen automatisk tilbakemeldingsgenerering og retting av tekstbaserte svar. I tillegg er det foretatt en undersøkelse på om teknikker for informasjonsgjenfinning og tekstdatautvikling kan brukes til å implementere et automatisk retting- og tilbakemeldingssystem, og hvilken kvalitet som kan oppnås.

En undersøkelse av eksisterende forskning om automatisk vurdering og generering av tilbakemeldinger er gjennomført, sammen med en oppsummering av relevante teknikker for informasjonsgjenfinning og tekstdatautvikling. To tilnærminger for automatisk vurdering og generering av tilbakemeldinger er foreslått og implementert. Den første omfatter en referansesvar (Reference Answer) basert tilnærming som sammenlignes med elevsvar ved hjelp av en likhetsberegning, og den andre benytter k-means-klynging (k-means clustering) algoritmen for å gruppere sammen like svar. Automatisk vurdering er forsøkt ved å utforske hvilke resultater som kan oppnås ved å bruke flere forskjellige informasjonsgjenfinningsteknikker (Information Retrieval techniques). Tilbakemeldingsgenerering foreslås ved å gi nøkkelord til elevene som representerer viktige begreper eller temaer de kanskje har utelatt i svaret sitt for å forbedre læringen. Videre er to metoder for å evaluere tilbakemeldingsteknikken foreslått og diskutert.

Denne oppgaven presenterer verdifull innsikt i hvordan norske og flerspråklige datasett kan renses og behandles, sammen med to lovende tilnærminger som kan vise seg nyttige og som kan brukes i et fullskala automatisk retting- og tilbakemeldingssystem i fremtiden.

Preface

This Master's thesis was written by Jesper Trøan and Julian Nyland Skattum during the spring of 2023 at the Norwegian University of Science and Technology (NTNU) as part of the Master of Science degree in Informatics.

Our sincere gratitude goes to our supervisor Trond Aalberg for all the excellent help and constructive feedback he provided throughout the work of this thesis. Finally, we would like to thank each other for our good cooperation during this project. It has been an interesting and educational process for us both.

Contents

Listings **vii**

List of Figures **viii**

List of Tables **ix**

1 Introduction **1**
- 1.1 Motivation/Background 1
- 1.2 Research Goals and Method 2
- 1.3 Contributions 3
- 1.4 Thesis Structure 4

2 Background **5**
- 2.1 CAA 5
- 2.2 Information Retrieval 6
- 2.3 Text Mining 7
- 2.4 Data Preprocessing 8
  - 2.4.1 Case Folding 9
  - 2.4.2 Tokenization 9
  - 2.4.3 Punctuation Removal 9
  - 2.4.4 Stopword Removal 9
  - 2.4.5 Stemming 9
  - 2.4.6 Lemmatization 10
  - 2.4.7 Additional Steps 10
- 2.5 Text Representation 10
  - 2.5.1 Bag-of-Words 11
  - 2.5.2 N-Grams 12
  - 2.5.3 Term Weighting 12
  - 2.5.4 TF-IDF 13
  - 2.5.5 The Term-Document Matrix 13
  - 2.5.6 Cosine Similarity 13
  - 2.5.7 Ranking Documents 13
  - 2.5.8 Vector Space Model 13
- 2.6 Latent Semantic Analysis 14
- 2.7 Evaluation Metrics 14
  - 2.7.1 Pearson Correlation 15
  - 2.7.2 Purity 15

2.7.3 Silhouette Score . . . 16
2.7.4 Chat-GPT . . . 16

3 Related Work **18**
3.1 History . . . 18
3.2 Evolution of Automatic Grading . . . 19
3.2.1 Concept Mapping . . . 19
3.2.2 Information Extraction . . . 20
3.2.3 Corpus-Based Methods . . . 21
3.2.4 Machine Learning . . . 22
3.2.5 Clustering . . . 23
3.3 Most Relevant Approaches . . . 25
3.3.1 Automatic Evaluation of Short Free-Text Answers . . . 25
3.3.2 Automatic Short-Answer Grading And Feedback Using Clustering Methods . 25
3.3.3 Automatic Evaluation of Students' Answers with LSA . . . 26

4 Design **27**
4.1 High-level Goal . . . 27
4.2 Comparing Student Answers with Reference Answers . . . 29
4.3 Clustering . . . 31

5 Dataset **33**
5.1 Data Cleaning . . . 34
5.2 Feature Analysis . . . 38
5.2.1 Grade Distribution . . . 39
5.2.2 Answer Length . . . 40

6 Implementation and Evaluation **43**
6.1 Tools and Environments . . . 43
6.2 Dataset Preparation . . . 43
6.2.1 Lexical Analysis . . . 44
6.2.2 Stopword Removal . . . 44
6.2.3 Stemming . . . 44
6.2.4 Lemmatization . . . 45
6.2.5 Synonym List . . . 45
6.2.6 Spell Correction . . . 46
6.2.7 Dimensionality Reduction . . . 47
6.3 Reference Answer Approach: Grading Results . . . 47
6.3.1 Test 1: All Highest-Graded Answers . . . 48
6.3.2 Test 2: Top 10 of the Highest-Graded Answers . . . 49
6.3.3 Test 3: Half of the Highest-Graded Answers . . . 50
6.3.4 Test 4: Most Common Terms in the Corpus . . . 50
6.3.5 Honorable Mention . . . 51
6.4 Reference Answer Approach: Feedback Generation Results . . . 51
6.5 Clustering Approach . . . 54
6.5.1 Grading Results . . . 57
6.5.2 Feedback Generation Results . . . 62
6.6 Code Quality . . . 66

7 Discussion 67
7.1 Dataset Reliability . . . 67
7.2 Reference Answer Approach . . . 68
7.3 Clustering Approach . . . 70
7.4 Limitations . . . 72

8 Conclusion 74
8.1 Conclusion . . . 74
8.2 Contributions . . . 75
8.3 Future Work . . . 76

Bibliography 79

A List of stopwords 83

B Exam questions 90

Listings

5.1 Snippet from processed JSON file 36
6.1 Find synonyms function 45
6.2 Example of Term-Document Matrix 46
6.3 Feedback generation function 52
6.4 Find optimal number of clusters function 55
6.5 Top N term extraction function 62

List of Figures

2.1 A simplified representation of retrieving matched documents 7
2.2 The NLP pipeline 8
2.3 Bag-of-words 11
2.4 N-grams text representation method 12
2.5 Term-Document Matrix 13
2.6 Chat-GPT response when prompting on Python stemming libraries 17

4.1 High-level goal 28
4.2 High-level feedback technique 28
4.3 High-level similarity approach 29
4.4 High-level clustering approach 31

5.1 Grade distributions 39
5.2 Word counts per grade in responses D18 41
5.3 Word counts per grade in responses D19 42

6.1 Overview of the clustering approach 56
6.2 Manualscores in different clusters D18 Q1 58
6.3 Manualscores in different clusters D18 Q6 58
6.4 TSNE plot D18 Q1 60
6.5 TSNE plot D19 Q6 61
6.6 TSNE plot D18 Q1 clusters highlighted 61

List of Tables

2.1 Interpretation of Pearson correlation coefficient 15

5.1 List of obstacles in the datasets 34
5.2 Languages in responses data D18 37
5.3 Languages in responses data D19 37
5.4 Feature analysis overview 38
5.5 Skewness and Kurtosis of datasets 39
5.6 Word Count Statistics by Grade D18 40
5.7 Word Count Statistics by Grade D19 40

6.1 Performance on all highly graded answers as Reference Answers 48
6.2 Performance on ten highly graded answers as Reference Answers 49
6.3 Performance on half of the highly graded answers as Reference Answers 50
6.4 Performance on most common words in the corpus used as reference 51
6.5 Feedback terms and Student Answers From Q1 on the 2018 dataset 52
6.6 Terms retrieved from feedback 53
6.7 Feedback precision 54
6.8 Purity scores for D18 Q1 and D18 Q6 59
6.9 Overall silhouette scores for clustering D18 Q1 and D18 Q6. 59
6.10 Silhouette scores for each cluster in D18 Q1 and D18 Q6 62
6.11 10 most frequent terms in each cluster D18 Q1 63
6.12 10 most frequent terms in each cluster D18 Q6 63
6.13 Feedback test 65

Chapter 1

Introduction

With the growing demand for higher education and the consequent increase in student enrollment, there exists a corresponding rise in the workload associated with evaluating student work. Evaluation of student text answers can be both challenging and time-consuming to perform, and teachers dedicate a substantial amount of their time evaluating such answers. In addition, it is difficult to ensure that all answers get evaluated fairly and in accordance with the evaluation criteria [38]. Furthermore, the increased workload also entails higher costs. To tackle this issue, there is a consensus among researchers that computer technology can be leveraged to aid in their assessment tasks [34].

The need for electronic assessment of student short answers grows as the replacements of handwritten assignments with digital deliveries increase. Assessing the answers, in addition to providing feedback to students, is beneficial to improve learning. These are key factors that have motivated research into the fields of automatic grading and feedback generation.

This thesis is a study in the field of automatic assessment of student short answers, with the focus on providing feedback to students to enhance learning. Two automatic grading and feedback generation systems are proposed and implemented with Text Mining (TM) and Information Retrieval (IR) techniques. Furthermore, they are applied and discussed on two Norwegian text-based exams.

1.1 Motivation/Background

The evaluation process can be time-consuming and mentally taxing for teachers. Maintaining consistent focus and concentration over extended periods is challenging, and variations in mood and attention can affect the grading process. As a result, answers of similar quality may receive different grades [38]. Furthermore, the order in which the teacher grades answers can cause discrepancies. Multiple low-quality responses might receive higher scores if high-graded responses have been graded beforehand, and similarly, high-grade responses might receive medium-quality scores if medium-quality responses were recently graded. Consequently, the motivation for a computer-based automatic evaluation system is evident, and this field of research is called Computer Assisted Assessment (CAA) [34]. Multiple CAA systems have been constructed and vary greatly in the way they assess answers and implementation techniques, such as style and/or content. The most successful approaches to text evaluation found in CAA systems are statistical, Latent Semantic Analysis (LSA) and Natural Language Processing (NLP) [38]. Initially, CAA systems were statistical and focused solely on the structural similarity of texts. LSA, however, introduced a comparison based on a corpus and an algebraic technique to calculate similarities between texts with various terms. The more recent advancements in NLP techniques can also retrieve the semantic meaning of free-text documents [38]. Presently, the advances within NLP have made the research in this field increase greatly, and computers can now analyze text answers in a better, more detailed, and efficient way than a teacher can without their corresponding biases [34]. However, since the techniques utilized and evaluation criteria for text evaluation differ, it is difficult to compare CAA systems. There is also a lack of common corpora and metrics within this field [34]. In addition, because the structure of text answers varies greatly, the performance of such metrics can as well.

Within CAA, the need for not only assessing text answers but also providing feedback to students to improve learning is important. It is a key factor in both learning and teaching. When students receive feedback, strengths and weaknesses can be identified, and potentially improved. In addition, this concept can help set realistic goals and develop strategies to achieve them. Additionally, it enables students to identify and resolve misunderstandings and improve their understanding of the subject. A well-performing CAA system should deliver quick, precise, and constructive feedback that supplies students with concrete steps to improve their performance and knowledge. In summary, providing feedback to students plays a considerable role to help them understand subjects, improve learning, and self-awareness [41].

Although grading text answers poses a time-consuming task for teachers, providing feedback presents an additional bottleneck due to the necessity for it to be individualized for each student. This process may prove complicated as it necessitates a thorough understanding and analysis of the student's answers [41]. Therefore, feedback generation should be integrated within a CAA system, whereas the same techniques for grading may be utilized. One approach for such a system is employing the principle of similarity between a Reference Answer (RA) and a Student Answer (SA). Such an approach can calculate the distance between two texts to find misconceptions and weaknesses in student responses, which can further be utilized as feedback to students and assess their level of understanding.

The deployment of CAA systems has the potential to completely change how academic assessments are conducted. As a result, teachers can grade a large number of responses simultaneously and almost instantly offer results in an accurate and more objective manner than a human evaluator. Consequently, the time needed for grading and assessing student responses shortens drastically when utilizing such a system. Furthermore, it ensures consistency and fairness in the evaluation process. Nevertheless, the absence of common metrics, corpora, and performance variations necessitates further research within this domain.

1.2 Research Goals and Method

The overarching research goal of this thesis is to study how Information Retrieval and Text Mining techniques can be utilized to evaluate short-text answers from students on two Norwegian exams in computer science. Furthermore, it is to look at how such techniques can be applied to generate feedback from answers to enhance learning in students. A thorough dataset analysis of the two datasets (exams) is presented, and two approaches for implementing such a grading and feedback system have been proposed, implemented, and evaluated. Furthermore, three following research questions have been formulated:

- RQ1 - What is state-of-the-art within automatic feedback generation and grading of text-based answers?
- RQ2 - How can Information Retrieval and Text Mining techniques be applied to implement an automatic grading and feedback system?
- RQ3 - How do automatic grading and feedback systems perform on real-world datasets?

To address the first research question, a literature study within the field of automatic feedback generation and grading has been performed. This analysis is presented as Background and Related Work chapters in this thesis and presents insights into earlier work and methods implemented within this field. The analysis covers existing solutions on feedback generation and grading systems, as well as explanations of important IR and TM techniques necessary to understand the existing solutions and proposed approaches in this thesis. Addressing the research question in such a way will provide vital knowledge to both perform and understand the dataset analysis presented. Furthermore, it will optimally aid in designing high-quality approaches for an automatic grading and feedback system.

The second research question will mainly be discussed in the Background chapter of this thesis, which covers several Text Mining and Information Retrieval techniques, as well as the preprocessing pipeline. In addition, the proposed approaches for implementing a feedback and grading system will be composed of such techniques. Exploring this research question will provide the reader of this thesis with insights into the fields of Text Mining and Information Retrieval, including realistic examples and approaches.

To approach the third research question, it is important that a dataset analysis is completed and performed with high quality to ensure the proposed approaches can be evaluated with high-quality data. Following the analysis, multiple experiments on different techniques must be carried out to evaluate the feedback and grading system, and to determine which approach is the most effective. Related work regarding other successful systems on this topic is highly relevant for this implementation. The goal will be to evaluate several Information Retrieval and Text Mining techniques on grading and feedback generation to determine how they perform on two Norwegian text-based exams. Furthermore, it is important that the feedback generated from these approaches are of high quality and measurable in some manner that supports the end goal, to enhance learning. Optimally, the approach and implementation of such a system should be built and explained in a manner that is easily understandable and utilized later.

1.3 Contributions

There is little to no work observed on feedback generation of text-based answers in the Norwegian language. Existing work on automated grading in Norwegian is also minimal. Consequently, the work presented in this thesis contributes to further research on feedback generation of text-based answers in the Norwegian language, as well as proposed approaches and implementations of future systems for feedback generation and grading. The main contributions of this thesis are listed below:

- A dataset analysis on two Norwegian text-based exams.
- An investigation into existing related work in the fields of automatic grading and feedback generation.

- Implementation and presentation of two proposed approaches for automatic grading and feedback generation.
- Performance evaluation of the two presented approaches on two Norwegian text-based exams.

1.4 Thesis Structure

The rest of this thesis is structured as follows:

Chapter 2 covers background information that is relevant for the reader of this thesis. This includes methods and technologies within Information Retrieval and Text Mining that are referenced and utilized throughout this work.

Chapter 3 introduces related work within the field of automatic grading and feedback. This includes a timeline and history of relevant literature in addition to relevant approaches for automatic grading and feedback.

Chapter 4 presents the two approaches for automatic grading and feedback. The chapter initiates with the high-level goal of this thesis, followed by a high-level design and description of the two approaches.

Chapter 5 covers an analysis of the two datasets. It introduces and explains the contents of the data and summarizes it to provide an overview. Subsequently, the performed procedure of data cleaning is explained, along with a discussion of vital decisions taken during this step. Various significant issues regarding the datasets are also presented along with a feature analysis.

Chapter 6 covers the implementation and evaluation of two approaches for automatic grading and feedback generation on the cleaned datasets. Furthermore, it elaborates on the relevance of these approaches in the context of current research and established techniques in Information Retrieval and Text Mining.

Chapter 7 discusses the findings from this thesis and the performance of the two proposed approaches on the datasets. In addition, possible pitfalls and limitations of these approaches are presented.

Lastly, **Chapter 8** initiates with revisiting the three research questions. Furthermore, it concludes this thesis and provides suggestions for future work to improve or continue the work performed.

A preliminary project [42] was conducted prior to the construction of this thesis. Consequently, sections in Chapters 2 and 3 have been conveniently reused and adapted to better fit the research objectives of this thesis.

Chapter 2

Background

2.1 CAA

The term CAA refers to all summative (tests that will contribute to academic qualifications) and formative (tests that support the learning process but are not part of a course's final grade) assessments of students' progress that are administered with the help of computers [11]. Online tests or quizzes are one of the most popular types of CAAs. These systems are designed to streamline and automate the assessment process, making it more efficient, scalable, and less subjective to human biases. Such systems are can support a variety of assessment formats, including:

- **Multiple-choice questions:** These systems provide students with a list of questions and a choice of answers for each question where only one option is correct.
- **Short-answer questions:** Such systems require the students to write a short and concise response to a given question.
- **Long-answer types:** These systems require students to write longer and more detailed responses, such as a thesis, essays and reports
- **Algorithmic problem-solving systems:** Such systems provide students with a problem that must be solved with the use of an algorithm the student must develop/solve.

The focus of this thesis is on short-answer responses. These responses are typically limited to a few words, a sentence, or a brief paragraph depending on the context and the requirements set by the assessment system or instructor.

In this thesis, we define a CAA system to include both grading and feedback generation. The grading system encompasses evaluating Norwegian short answers from students and assigning a grade to the given short answer. Giving grades in CAA systems often depends on how accurate and complete the student's response is. A common approach is to compare the student's answer to a pre-defined list of correct responses or teacher-provided Reference Answers. Such a system can employ different techniques to compare the student's answer with a Reference Answer, such as semantic similarity matching or statistical methods [37]. Based on how closely the student's response is to a Reference Answer, the system gives a score, and this score is used to calculate a grade. A CAA system may also consider additional features in the grading process, such as the length of the response or the presence of specific concepts or ideas. Furthermore, it can be customized based on the learning goals of a subject and developed to provide immediate and objective feedback to students. Throughout this thesis, the focus will be on two proposed approaches for grading and feedback generation in a CAA system. One approach is based on finding the similarity between a Student Answer and a Reference Answer, and the other is based on grouping similar responses, also known as clustering.

The feedback generation in such a system should be appropriate to the grade. There exist numerous ways of presenting feedback. One such type is a text that provides information on the partial correctness of the answer. For instance, the system can offer feedback on the specific part that is correct and the part that needs improvement. Another type is feedback which focuses on helping the students reflect on their learning process and strategies. This may be achieved by providing feedback on the effectiveness of their response or by encouraging self-assessment, which can assist the students in reflecting on their own strengths and weaknesses.

In the context of this thesis, the term ‘feedback’ is meant as presenting keywords to students based on their responses to short-answer questions. These keywords are terms representing important concepts or themes that are not present in their short-answer response. The goal is, in combination with automatic grading, to present meaningful feedback in the form of keywords to enhance learning and improve self-awareness in the students learning of short-text answers. Information Retrieval techniques, data preprocessing, and ranking functions are essential parts of CAA systems that will be applied in the attempt to achieve this goal.

2.2 Information Retrieval

Information Retrieval (IR) is the process of retrieving information from a collection of documents, images, videos, or other forms of data. It is based on a user’s query, and encompasses developing algorithms and methods for searching, manipulating and organizing large amounts of information effectively and efficiently [49]. The goal of IR is to provide relevant and accurate information in response to the user’s query, thus, making it a major area of research within computer science. IR involves a range of techniques and algorithms such as text-based indexing and relevance feedback, whereas the success of these systems depends on the accuracy of the algorithms, as well as the quality of the data used. In the context of CAA systems, IR can be utilized to retrieve data from various sources to support the assessment process. For instance, student records and test results can be retrieved and manipulated for further learning and evaluation.

The applications of IR systems can be found in various domains, with web search engines standing out as particularly intuitive. In today’s world, IR systems are essential due to the constantly growing amount of information stored and generated. Consequently, the challenge of retrieving and finding relevant information for users increases. The primary goal of an IR system is therefore to retrieve all relevant documents from a collection whilst retrieving as few non-relevant documents as possible [3]. Furthermore, ranking documents based on their relevancy to a user’s query is often employed to determine their importance.

The queries given by a user in an IR system are usually strings containing keywords or phrases of the information the user desires to retrieve. Several documents in the collection may match this query, usually with varying degrees of relevance. This relevance is normally based on a similarity calculation between the input query against all the documents in a collection. The similarity score is then used to rank these documents. Before such a similarity score is calculated, however, the query and the documents need to be represented in a similar manner. This makes measuring similarity possible. Figure 2.1 presents an illustration of how an IR system can retrieve documents from a document collection based on a user's query.

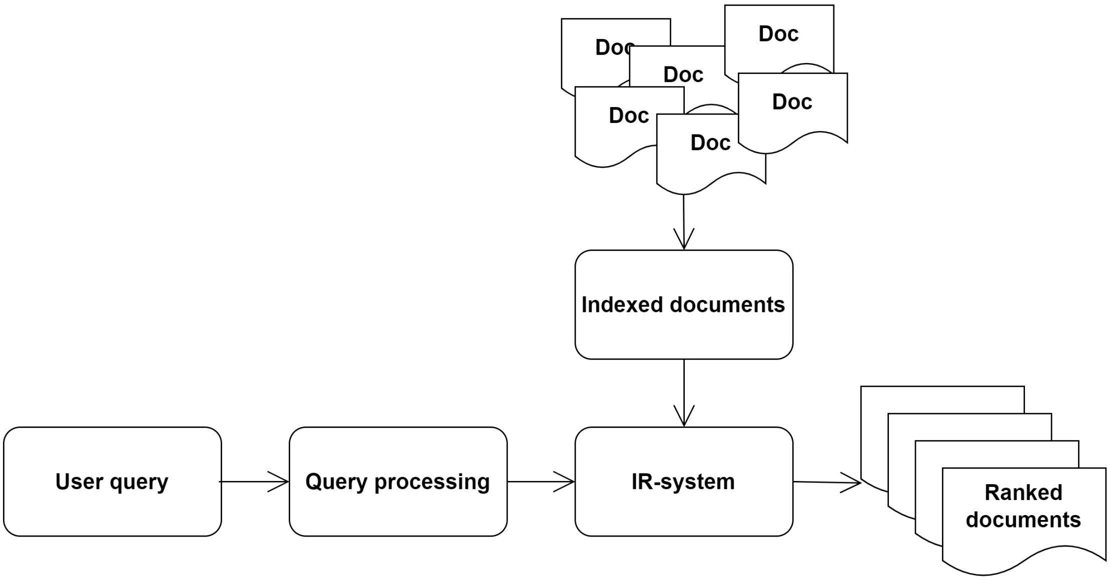


Figure 2.1: A simplified representation of retrieving matched documents

In CAA systems, Information Retrieval is one of the main parts utilized to compare Student Answers (SAs) with one or more Reference Answers (RAs) to find potential similarities. The goal of an IR system is, as mentioned, to retrieve relevant documents from a large corpus of text based on a query. In the case of a CAA system, the large corpus of text can be a set of one or more RAs, and the query is a SA that requires grading. One challenge in using IR for short-answer grading is the issue of vocabulary mismatch, where the students select different words or phrases to describe the same concept as the Reference Answers. To compensate for this issue, techniques such as synonym matching and paraphrase detection can be applied to find semantically similar solutions.

2.3 Text Mining

Text Mining (TM) is a field that encompasses the automatic extraction of meaningful data from large collections of unstructured or semi-structured text. It involves computational techniques that extract such data. The goal is to retrieve valuable insights from data by using a range of techniques. Such techniques may include text preprocessing, information extraction, and document clustering [50]. By incorporating such techniques in the form of algorithms, it becomes possible to retrieve facts, relationships, and assertions that would otherwise remain undiscovered in the mass of textual big data. When this data is extracted, the information is usually converted into a structured format for further analysis.

TM techniques can be utilized in a wide variety of domains. In business, market trends and competitor analysis can be constructed by the utilization of such techniques. Within social media, it can be employed to precisely analyze the number of posts, followers, and likes from an individual. Furthermore, TM techniques can be useful in scientific research to extract valuable research from collections of scientific literature, such as publications, patents, and conference proceedings.

Although the fields of TM and IR are closely related, they appear as two distinct fields of study. IR focuses on retrieving information from document collections or data sources, while TM focuses on the automatic extraction of meaningful and useful information with a focus on transforming raw text data into a structured and meaningful manner.

2.4 Data Preprocessing

In order to evaluate text answers automatically, it is important to translate the unstructured text into a language that the computer can understand. The process of transforming text into a format that can be fed into a model for further analysis is called text preprocessing and is often referred to as Natural Language Processing (NLP). The goal of text preprocessing is to reduce the amount of irrelevant information and noise from the text, as well as extract the relevant and specific information that is needed for a certain task. It is a crucial step in NLP because the quality of the model for analysis highly depends on the quality of the data it is given. Thus, a text preprocessing pipeline that is well-designed can improve the performance of IR and TM systems. Several steps and techniques exist to perform text preprocessing, and among the most popular are lemmatization, stemming, and tokenization. Figure 2.2 illustrates what a text preprocessing pipeline could look like.

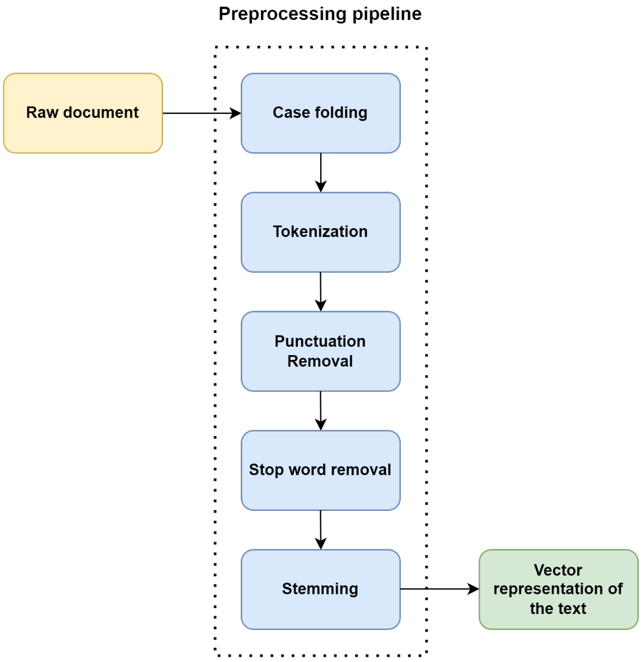


Figure 2.2: The NLP pipeline

2.4.1 Case Folding

Case folding is a step where the text is changed to a standard case, generally lowercase. With case folding, the vocabulary size can be reduced and, consequently, lead to a dimensionality reduction of the data. This allows for better generalization of the NLP pipeline.

2.4.2 Tokenization

Tokenization is the process of breaking a stream of text into pieces, called tokens. Tokens are usually a single term, or a sequence of terms that are grouped together, having a useful semantic meaning for processing. It is a crucial step in the NLP pipeline as it allows for text to be represented as meaningful units, causing it to be easier to process and analyze. Such tokens can be used directly as a vector representing specific documents. It is normal to break down the stream of text by splitting it on whitespace or into subwords (n-grams), whereas the choice of split depends on the task and type of text that is being processed. The main goal of tokenization is to split the text into tokens that preserve the meaning that was intended in the text.

2.4.3 Punctuation Removal

Punctuation removal is the process of removing punctuations, or special symbols. This process is often performed together with the tokenization step. The punctuation removal process will aid in treating each individual text equally. For example, the words 'Food' and 'Food!' are treated equally after the punctuation removal process. It is important to select which punctuations to remove with care in order to preserve the intended meaning of the text. Although this process generally increases the performance of TM algorithms, the opposite can occur if the punctuations are not deliberately selected. Periods, commas, semicolons, and colons are examples of often removed punctuations.

2.4.4 Stopword Removal

Stopword removal is one of the most commonly used preprocessing steps in NLP. The definition of a stopword is a word that has little to no significant semantic context or is commonly used across the corpus. Such words often have little contribution to the meaning of the text. Typically, articles and pronouns are generally classified as such words. Examples are words like 'the', 'a', 'an' and 'in'. Usually, a list of stopwords for a given language is a hand-curated list of words that occur most commonly across the corpus. The removal of stopwords can reduce the dimensionality of the data, and consequently, the resulting text can be simpler to process.

2.4.5 Stemming

Stemming is the process of removing a part of a word to its 'stem', or root by removing its affixes. This way, words with a common origin can be combined. As an example, assume the words: 'write', 'wrote', and 'written'. These words are all different forms of the same word 'write'. After stemming is performed, three occurrences of the word 'write' is present. A handful of different algorithms are used to decide how to 'stem' a word, and sometimes the algorithms will stumble into some complications. These complications are called *over-stemming* and *under-stemming.* To put it simply, *over-stemming* is the case where a much larger part of a word is chopped off than what is required. This causes two or more distinct words to get reduced to the same root word incorrectly. For instance, 'Anticlimax' and 'Antisocial' are words of different meanings, but an algorithm that is subject to *over-stemming* may replace both words with the same 'stem'. *Under-stemming* occurs when two or more words are wrongly reduced to more than one root word when they ideally should be reduced to the same root word. Overall, the point of stemming is to increase the performance of different NLP tasks by reducing the number of words in the corpus. This way, words of the same meaning may be matched together by their stem.

2.4.6 Lemmatization

Lemmatization is the process of reducing words to their base or dictionary form, called lemma. It involves determining the context of a word and from there deriving its base form. It differs from stemming because it uses the context of a word to obtain its canonical form. Stemming, however, only removes a word's affixes to build a stem. As an example, consider the words 'am', 'are' and 'is'. these words would be lemmatized to the lemma 'be', but the stemming process might produce different stems from the three words. In order to use or create a lemmatizer, a list of words or a lexicon is required. Such a lemmatizer should contain as many words as possible to match the dictionary form.

The goal of both lemmatization and stemming is the same, to reduce words to a common form in order to reduce the number of words in a corpus, and consequently, reduce the dimensionality. The discussion of which process to use varies on the task. However, a combination of both has been shown to give a high performance. Generally, lemmatization should be used in context analysis, and stemming should be used when the context is not important [44].

2.4.7 Additional Steps

The preprocessing steps mentioned above are somewhat general and are widely utilized for a series of tasks. However, additional steps can be taken to further preprocess the text. These steps are dependent on the task and may include language identification, part-of-speech tagging, and further text cleaning. Language identification encompasses attempting to identify the language in which a document is written. For the majority of tasks, it is important that all documents are in the same language for a TM algorithm to function properly. Thus, removing documents of an unwanted language can be performed. Such a preprocessing step can be applied in language translation applications or multilingual search engines.

Part-of-speech tagging is the process of labeling each word in a text with its part-of-speech, for example, noun, verb or adjective. This can be useful for understanding the grammatical structure of the text. Further cleaning of a text can also be relevant if the text contains more unwanted characters, such as non-alphabetic characters or HTML tags.

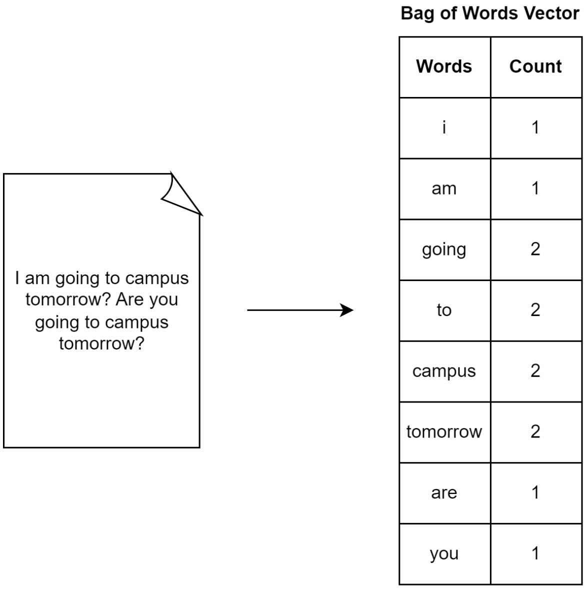
2.5 Text Representation

After the raw document is run through the pre-processing pipeline, the need for representing the text in an understandable manner for computers comes into place. Natural Language Processing (NLP) is a subfield of artificial intelligence and can make machines understand and process human language [10]. Feature extraction is a general term that encompasses transforming the raw document into numerical features that can be processed while preserving the information from the original dataset. This is necessary so that machines can understand and identify human language.

Typically in an IR system, a document is represented by index terms. An index term may refer to any word found within the document or to tokens generated after the preprocessing step. The set of distinct terms in a document collection is referred to as the vocabulary V of the collection. This vocabulary is important as it identifies all index terms in the collection. It can grow quickly depending on how big the document collection is, and consequently, preprocessing steps can be performed to minimize this. Within Text Mining, text representation is one of the largest problems. Multiple approaches have been proposed to model text in such a way that computers can understand it, and this section presents some of the most relevant models as well as similarity and ranking measures.

2.5.1 Bag-of-Words

Bag-of-words is one of the most popular and used text vectorization techniques. Furthermore, it is a common approach to utilize IR in short-answer grading. Here, a document is represented as a 'bag' of words that describes the occurrence of each word in the document. It includes both a vocabulary of all the words within the document and a count for each word's occurrence. The reason that this text representation method is called bag-of-words is that any information regarding the order or structure of words in the document is ignored [7]. To sum it up, this model singularly provides information about the words that are present in the document and their frequency with no regard for word order or context. The result of such a technique is a vector of word counts used to represent the text. Figure 2.3 provides an illustration of this text representation technique.



I am going to campus tomorrow? Are you going to campus tomorrow?

Bag of Words Vector

Words	Count
i	1
am	1
going	2
to	2
campus	2
tomorrow	2
are	1
you	1

Figure 2.3: Bag-of-words

2.5.2 N-Grams

N-grams and bag-of-words are similar in several ways. N-grams are essentially a collection of words that frequently appear in a certain window. A group of N words, to put it another way. The value of n is decided by the user. The term unigrams is used when N=1. Tokens in a unigram are basically individual words in the text. Bigrams and trigrams are the terms used when N=2 and N=3 respectively. N-gram models are often used to estimate the probability of the last word given the previous words [27]. As an example, assume the text ‘Please drink your’. The probability of the next word being ‘water’ is higher than the probability of the next word being ‘spoon’.

N-grams are also used in Information Retrieval with the aim of finding similar documents given a single query and a set of documents. Some advantages of this are that the model is more capable of capturing the semantic meaning of the sentence. If the user uses bigram or trigram which is a sequence of words, it gets easier to find the word relationships. It is also easy to implement as it requires few modifications from bag-of-words. However, one disadvantage when not using unigram is that the model produces a larger, sparser feature set. Figure 2.4 illustrates how to use n-grams to represent the text ‘We are two students trying to explain N-grams’.

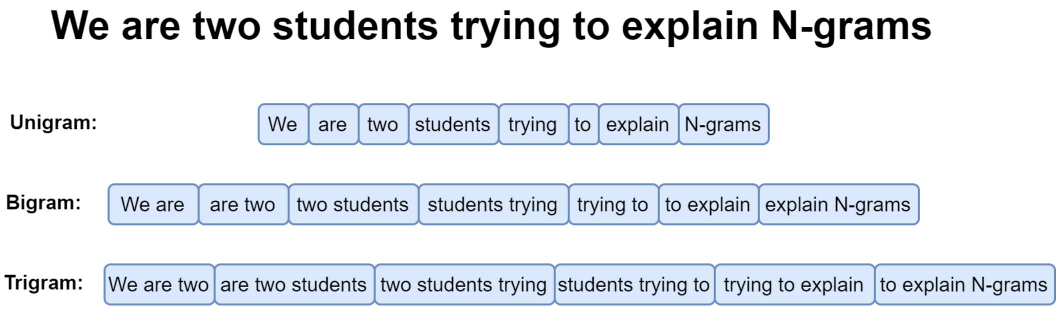


Figure 2.4: N-grams text representation method

2.5.3 Term Weighting

Term weighting is a method of expressing the presence of a token or term in a corpus. The weight of the term represents the importance of it in a document and is used to improve retrieval effectiveness. There exist multiple methods of weighting terms, one of which is the boolean model. The boolean model simply assigns a weight of 1 if the term is present in the document, and 0 if it is not present. This causes all present terms to have the same weight, which might not be the best representation because some terms could have bigger importance.

Term Frequency (TF) is another weighting method that counts the number of times a term occurs in the document. This causes the term to have a higher weight the more it is mentioned in the document. However, this method can be rather misleading and does not guarantee high retrieval performance because there are very common words (like stopwords) that appear in the text, but carry little meaning [24]. This approach also causes documents of longer length to be favored because terms might appear more often in longer documents than in shorter ones. To tackle this, *sublinear tf scaling* can be utilized.

2.5.4 TF-IDF

To counter the drawbacks of only using the raw TF, each token can instead be assigned a weight indicating the importance in the document (TF), as well as its relative importance in the whole document collection (IDF). The Inverse Document Frequency (IDF) is a measure of the rareness of a given term in a document collection. By combining the TF and IDF, a statistical measure is constructed to evaluate the relevance of a term to a document in a document collection. The TF-IDF approach is, according to Baeza-Yates et al. [3] the most popular term weighting model in IR.

2.5.5 The Term-Document Matrix

The occurrence of a term in a document causes a relationship between them, and after representing the text in some manner, the resulting model can be exploited using a matrix structure. The Term-Document Matrix (TDM) represents the full document collection and all terms in the vocabulary. Given a vocabulary V and a document collection D, the matrix can be described as in Figure 2.5 where $w_{i,j}$ represents the weight of term k_i in document d_j.

$$\begin{array}{c} \\ k_1 \\ k_2 \\ k_3 \end{array} \begin{array}{c} \begin{array}{cc} d_1 & d_2 \end{array} \\ \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \\ w_{3,1} & w_{3,2} \end{bmatrix} \end{array}$$

Figure 2.5: Term-Document Matrix

2.5.6 Cosine Similarity

Once the Term-Document Matrix has been built, similarities can be calculated between documents. The cosine similarity method is used to measure the similarity between two texts regardless of their lengths. The equation is presented below:

$$cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{te}}{|\mathbf{t}||\mathbf{e}|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \tag{2.1}$$

2.5.7 Ranking Documents

Relevance ranking is one of the most important features of the Information Retrieval system. Ideally, the user only wants to retrieve documents that are as close as possible to the initial query. To achieve this, good ranking functions are essential. These ranking functions are explained in Sections 2.5.3, 2.5.4, and 2.5.6. The text representation models explained in Sections 2.5.1 and 2.5.2 together with the ranking functions provide the necessary tools for creating ranking models and ranking documents that are similar to a given query. The Vector Space Model (VSM) is one example of a ranking model that can be utilized for this purpose.

2.5.8 Vector Space Model

The Vector Space Model (VSM) has been in the IR research field since the 1960s and 70s and is still used frequently. It provides a simple and intuitive framework for implementing both term weighting, ranking, and relevance feedback. Both the documents and queries are viewed as vectors in a 3-dimensional space. The process of transforming these queries and documents into vectors is called text vectorization. The most popular text vectorizer is TF-IDF, which is explained in Section 2.5.4. The whole model is based on one basic idea: If document one is more similar to the query than document two, then document one is more relevant than document two. The model computes the similarity between the query and each document in the collection and based on this, the model ranks the documents.

2.6 Latent Semantic Analysis

Latent Semantic Analysis (LSA) was first introduced as a technique for improving IR. The key idea was to reduce the dimensionality problem in Information Retrieval. Most approaches to retrieving information are to do a lexical match between a user query and documents. This causes a large amount of irrelevant information to be retrieved when searching. LSA features a statistical approach to extract relations among words and documents. It analyzes the relationships between documents and the terms they contain by producing concepts related to the documents and terms. A Term-Document Matrix is built to find similarity structures, where rows are terms and documents are columns, and each individual cell entry contains the frequency at which the term occurs in the document. The matrix is often transformed in a sublinear fashion to provide a better performance, usually an IDF or entropy-based score [12].

After the Term-Document Matrix is constructed, a dimensionality reduction on the matrix is normally performed. Singular Value Decomposition (SVD) is performed to retain the k largest values in the matrix, and the remainder is set to 0. This creates an approximation from the original matrix, and similarities are computed within this dimension-reduced space. In the matrix, since both the terms and documents are represented as vectors, it is easy to calculate similarities between document-document, term-term, and term-document. Clustering can also be performed within this vector space by finding documents similar to a query and creating a new query vector built as a centroid of the similar term/document vectors within the cluster. The cosine similarity calculation between vectors is often used to measure the similarity in such an approach and has proven to be effective in many Information Retrieval applications [12].

Although an LSA approach can be implemented purely statistically, a combination of LSA and NLP techniques is often used to tackle the dimensionality problem in a better manner. Stemming, explained in Section 2.4.5 is broadly used to convert words to their morphological root, and controlled vocabularies are also an efficient way to limit variability in the query by having a requirement of an indexed set of terms. Full-text indexing is also an alternative to controlled vocabularies because assigning terms in a controlled vocabulary can be very time-consuming. However, a combination of both is often used to increase performance.

2.7 Evaluation Metrics

CAA systems should be evaluated to determine how effectively the system performs. Evaluation metrics are used to measure that statistical or machine-learning models are operating correctly and optimally. It is also important to use multiple evaluation metrics to evaluate a model. This is because a model may perform well when using one evaluation metric, but badly when using a different one. A few popular metrics used in evaluating CAA systems are discussed in this section.

2.7.1 Pearson Correlation

The statistical measure of Pearson correlation [43] indicates the strength and direction of the linear relationship between two variables. It is a frequently used method for assessing the strength of the connection between variables. The Pearson correlation coefficient ranges from -1 to +1. A value of -1 depicts a fully negative correlation, 0 indicates no correlation, and a value of +1 indicates a perfect positive correlation. This implies that the closer the correlation is to 1 or -1, the stronger the linear relationship between the variables is. Interpretations of the relationship strength may vary between disciplines, but there exist certain general rules of thumb [43]. This is illustrated in Table 2.1.

Table 2.1: Interpretation of Pearson correlation coefficient

Pearson correlation coefficient value	**Strength**	**Direction**
> 0.5	Strong	Positive
0.3–0.5	Moderate	Positive
0–0.3	Weak	Positive
0	None	None
-0.3–0	Weak	Negative
-0.5–-0.3	Moderate	Negative
< -0.5	Strong	Negative

In a CAA system, the Pearson correlation can be utilized to calculate the strength of the linear relationship between a cosine similarity (Section 2.5.6), which tells how similar a response is to a Reference Answer, and the grades given by a teacher. The higher the Pearson correlation score is, the more the CAA system agrees with a teacher, and an argument can be given that a Pearson correlation score close to 1 or -1 may indicate a more accurate score provided by the CAA system.

2.7.2 Purity

Purity is a metric used to assess how well clustering algorithms perform. To calculate the purity of a clustering result, a label is assigned to each cluster based on the most frequent class in it. The purity is then calculated as the number of correctly matched class and cluster labels in each cluster divided by the total number of data points [52]. Purity ranges from 0 to 1, with a value of 1 indicating that all items in each cluster are members of the same class. Equation 2.2 presents how purity is calculated for the entire clustering scheme. In this equation, N is the total number of data points, k is the number of clusters, m is the number of classes, C_i is the ith cluster, and T_j is the jth class. The formula determines the maximum number of items belonging to the same class in each cluster, adds these maximums across all clusters, and then divides them by the total number of items to determine the proportion of correctly assigned items.

$$\text{Purity} = \frac{1}{N} \sum_{i=1}^{k} \max_{j=1}^{m} |C_i \cap T_j| \tag{2.2}$$

Purity is an easy-to-understand metric for clustering performance, but it has several drawbacks. Purity can be misleading if the different classes in the dataset are unbalanced, and does not take into account the structure or density of the data. In general, the purity score increases as the number of clusters increases. For instance, the purity becomes one if a model that groups each observation into a separate cluster is used. Because of this, purity cannot be used as a trade-off between the number of clusters and clustering quality [52]. As such, purity is best used together with other metrics, such as silhouette score.

2.7.3 Silhouette Score

Silhouette score is also a metric used to evaluate the quality of clustering results. This metric measures how well each data point in a cluster fits with other data points in the same cluster compared to data points in other clusters. The score ranges from -1 to 1, whereas a score of 1 indicates that clusters are well apart from each other and clearly distinguished, while a score of 0 indicates that clusters are indifferent, or the distance between the clusters is not significant. A score of -1 means the clusters are assigned in the wrong way [5]. The calculation of the silhouette score is presented in Equation 2.3. Here, i is the data point being evaluated, $a(i)$ is the average distance between i and all other data points in the same cluster (average intra-cluster distance), and $b(i)$ is the minimum average distance between i and all data points in any other cluster (average inter-cluster distance). The average silhouette score for all the data points in a cluster determines the cluster's silhouette score, and the overall silhouette score for a clustering approach is the average silhouette score of all data points in all clusters.

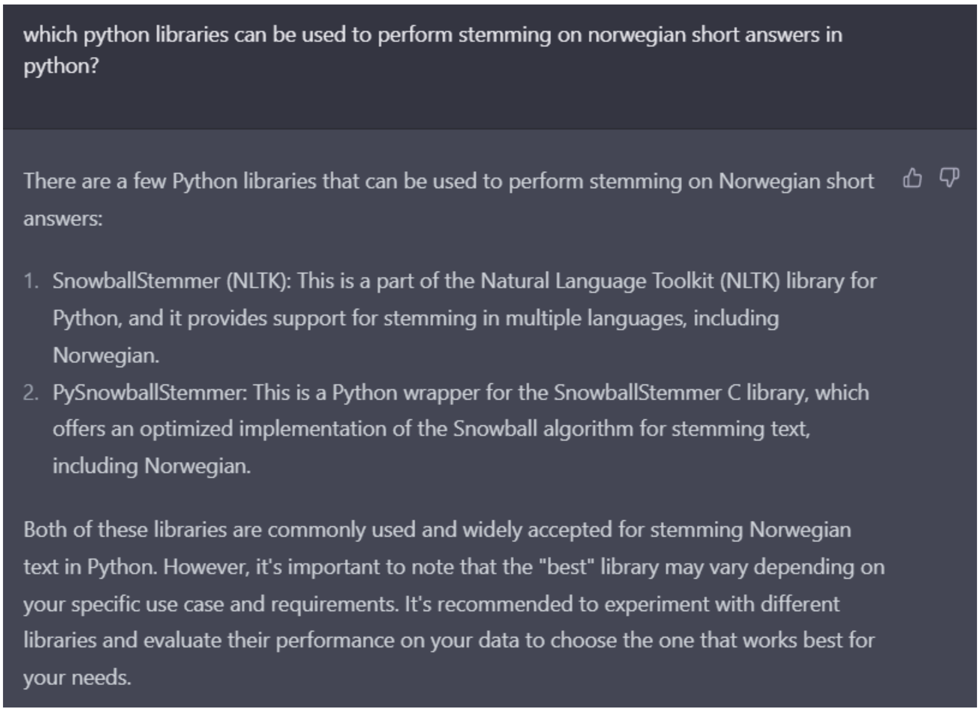
$$Silhouette(i) = \frac{b(i) - a(i)}{\max[a(i), b(i)]} \tag{2.3}$$

The silhouette score is a valuable metric for figuring out how many clusters are best suited for a particular dataset as it offers an objective evaluation of the quality of clustering that can be used to evaluate various clustering methods. However, a few limitations are present for this metric as well. The silhouette score assumes that the data is well-clustered and functions poorly with overlapping clusters or clusters with different densities. There also exist different distance metrics that can be used in the silhouette score calculation, and the distance metric used can impact the results of the score. Some of the most popular distance metrics are the Euclidean distance and the Manhattan distance.

2.7.4 Chat-GPT

Chat-GPT, a language model developed by OpenAI is an artificial intelligence (AI) chatbot released in November 2022. Chat-GPT is developed on top of the OpenAI's GPT-3.5 and GPT-4 families of large language models (LLMs). These language models consist of a state-of-the-art learning model that employs a transformer-based architecture with self-attention mechanisms to process and generate text [35]. GPT-3.5 is an extension of GPT-3, which is currently one of the most advanced language models in the world with 175 billion parameters [2]. It can perform various NLP tasks such as summarization, translation and composing emails and essays. GPT-4 was released in March 2023 and is even more powerful than GPT-3.5 with the number of parameters estimated to be in trillions [2]. This means that chat-GPT is trained on a massive amount of text data which enables the model to learn from diverse sources and from this generate contextually relevant responses in real-time.

Chat-GPT has gained much attention lately for its advanced text generation capabilities and ability to engage in dynamic conversations. Both models have already had a significant impact on the field of NLP and Artificial Intelligence (AI) area. Because of this, Chat-GPT can be used as a tool in this thesis. Examples are asking about different programming difficulties, relevant theory, and explanations of different functions and error messages. An example is presented in Figure 2.6 where the user wants to know about Python libraries that can stem short answers in Norwegian. However, it is also important to mention that there exist some concerns about the ethical implications, biases and limitations of utilizing Chat-GPT.



which python libraries can be used to perform stemming on norwegian short answers in python?

There are a few Python libraries that can be used to perform stemming on Norwegian short answers:

1. SnowballStemmer (NLTK): This is a part of the Natural Language Toolkit (NLTK) library for Python, and it provides support for stemming in multiple languages, including Norwegian.
2. PySnowballStemmer: This is a Python wrapper for the SnowballStemmer C library, which offers an optimized implementation of the Snowball algorithm for stemming text, including Norwegian.

Both of these libraries are commonly used and widely accepted for stemming Norwegian text in Python. However, it's important to note that the "best" library may vary depending on your specific use case and requirements. It's recommended to experiment with different libraries and evaluate their performance on your data to choose the one that works best for your needs.

Figure 2.6: Chat-GPT response when prompting on Python stemming libraries

Chapter 3

Related Work

Research on automatic grading and feedback of natural text has attracted researchers for a long time, and can be dated back to the early work done by Page in the 1960s [31]. Consequently, there exists a vast amount of previous and related work done in this field. This chapter presents some of the most relevant related work by providing a summary of the history of research in this field, as well as the most relevant literature for this thesis. This chapter is primarily meant to describe the state of the art within automatic grading and feedback generation to this date. Furthermore, it will provide insight and knowledge to further understand the work performed in this thesis.

3.1 History

Two popular question types to address learning outcomes in tests and exams are simple multiple-choice questions and questions requiring the learner to type in an answer in a natural language format. Examples of such formats are short answers and essays. Automatic Short-Answer Grading (ASAG) encompasses assessing short natural language responses to objective questions using computational methods [8]. The research regarding short-answer grading has increased rapidly over the last decade, with over 80 papers fitting the definition of ASAG [8]. Grading natural language responses and short-answer questions is more difficult than simply grading multiple-choice questions, as an understanding of the natural language is required.

Some of the earliest research done in this field is the work done by Page in 1966 [31]. With 276 essays submitted by high school students, Page conducted several experiments with automatic grading of student essays. These essays were then judged by four English teachers for overall quality. The ratings for each essay were added together and became the criterion of quality. Then, the computer was given variables that had the highest correlation with the actual ratings given by the instructors. These variables were identified by the teachers. Computer routines were written that analyzed these variables, performed the appropriate transformations, and produced a set of scores for each essay. The essays themselves had to be typed literatim on a key-punch machine for input into the computer, and when this was done, the program was executed. Results showed that grades computed by the computer were not distinguishable from human grades. With these results, Page observed that computerized grading can lead to the possibility of automated grading systems for the evaluation of essays in the future.

Since then, computerized grading of student responses has grown in popularity, and more substantial research has been conducted as a result of computers becoming more potent and gaining significant momentum in the 1990s and 2000s. Since Page's work, the methods have also diverged according to the type of question, such as short replies in comparison to essay answers. This thesis, however, focuses on short answers.

3.2 Evolution of Automatic Grading

One can observe that the existing ASAG systems have a few similarities based on their time periods of publication. In this section, different papers are divided into categories, or 'era's' in the field of ASAG. As stated by Burrows et al. [8], the historical papers can be divided into the four dominant era's 'Concept mapping', 'information extraction', 'corpus-based methods', and 'machine learning'. Furthermore, this section will present a clustering era where a set of clustering approaches will be discussed. For each era, a summarization of the key ideas and presentation of some of the most important papers will be conducted.

3.2.1 Concept Mapping

Concept mapping encompasses thinking that the students' answers are made up of different concepts, and the goal is to detect the presence or absence of each concept when attempting to grade. However, there is one limitation to this approach: only questions that ask for a solution to a problem plus a justification, or a question that asks for multiple explanations of a problem can be applied. In other words, the questions must accommodate the possibility of presenting concepts in the answers.

Burstein
Some of the earliest research regarding this era is the work of Burstein [9] which considers hypothesis-style questions. Here, multiple explanations must be given for a specific hypothesis. These explanations were each considered as an individual concept. A domain-specific, concept-based lexicon and a concept grammar were built to represent the response set using 200 of 378 responses from the original dataset. From this, concept grammar rules were developed by mapping concepts from the lexicon onto the concept-structure patterns present in a set of training responses. With this approach, the system managed to achieve 81% accuracy when assessed on the test set.

Wang
The work of Wang [46] aims to improve the assessment of creative problem solving by employing language technologies and computational-statistical machine learning methods to grade student answers automatically. This is done by comparing three methods for grading earth science questions in secondary education. These methods are based on concept mapping, machine learning, or both. The first concept mapping method is based on calculating cosine similarity on TF-IDF (Section 2.5.4) weights of bag-of-words features (Section 2.5.1). The second concept mapping method utilizes a Support Vector Machine (SVM) in addition to bag-of-words features. Furthermore, it is also implemented with machine learning and is therefore considered a combination of concept mapping and machine learning. The third method uses SVM regression with unigrams, bigrams, and part-of-speech bigrams. This method is purely machine-learning based. In contrast to the preceding two methods, the third method grades holistically and treats each concept as a single response.

C-rater
C-rater [25] is an automated scoring engine that was developed by ETS Technologies to score content-based short-answer responses in 2003, and is based on finding concepts within the students' responses. Prior to 2003, most research on automatic grading had focused on scoring essay-length responses and not short-answer questions. C-rater differs from essay scoring systems in several fundamental ways. It initiates with a model of the correct answer to a question that is created by an expert in the corresponding field of the question. C-rater will then map the student's response onto the model to demonstrate the correctness or incorrectness of the answer. This mapping is a fully automated process, unlike the construction of the model. Since C-rater utilizes a model that is designed to accommodate questions with either a single correct response or a range of correct answers, the questions must have a single valid response or a range of correct responses. One drawback of this is that open-ended questions, such as those that request examples from personal experience or opinions, cannot be handled.

To score short-answer questions, the scoring engine must be able to recognize concepts in the answers. C-rater analyzes responses using a range of NLP techniques and normalizes across the variety of ways a single concept can be expressed by focusing on four primary sources of variation in sentences: predicate-argument structure, pronominal reference, morphological analysis and synonyms. When a student's response has been normalized, C-rater will attempt to match the concepts it has identified to the concepts that are represented in the model of the correct answer, and then assign a score depending on the number of concepts that are matched. This system was tested and evaluated in two large-scale assessment programs: National Assessment for Educational Progress (NAEP) and a statewide assessment in Indiana. The C-rater system agreed with the human graders in both studies 84% of the time, which is a high correlation.

3.2.2 Information Extraction

In simple terms, Information Extraction (IE) involves taking data and extracting structured information to use it for a variety of purposes. IE automatically retrieves structured information from a set of unstructured documents or corpus [14]. In the context of this report, IE will encompass extracting facts/information from students' short answers. Usually, student answers contain specific ideas and facts that are beneficial to extract and model.

AutoMark
AutoMark (Mitcell et al. [29]) employs techniques of Information Extraction to provide computerized marking of short free-text student answers without penalizing the students in the case of errors in spelling, typing syntax and semantics. It looks for specific content in the short-text answers. This content is specified in the form of a number of mark scheme templates. Each template represents one form of a valid or invalid answer. When AutoMark grades an answer, the student answer is first parsed, then matched against each mark scheme template. Lastly, a mark for each answer is computed. These templates are designed such that they can robustly be mapped to multiple different variations of the input text. Two experiments were devised to evaluate the software of the system, namely the 'blind' and 'moderation' experiments. The blind experiment was a fully automated process, while the moderation experiment includes a step where humans could step in and revise after grading has been done.

eMax

eMax [39] is aimed at semi-automated assessment of short-text answers. The system initializes with lecturers entering the question to be asked and the expected correct answers into the system. In the freely formulated correct answer, the lecturer also needs to mark relevant semantic elements that are expected to be included in student answers. Following this, eMax offers a set of synonyms for each of the relevant semantic elements in the correct answer, and the teacher must then either accept or reject the synonyms offered. Finally, the lecturer must provide a scoring scheme for student answers by allocating weights to the semantic elements. With all these preconditions in place, eMax can perform an evaluation of the student's answers. Furthermore, the system also generates all possible formulations for each answer type when evaluating, and each assigned score is additionally given a confidence rating. This is performed so that challenging cases can be forwarded to a lecturer for manual review.

Auto-Assessor

The primary focus of the Auto-Assessor [13] system is to determine the semantic meaning of student responses, and a goal the developers of the system had in mind was that the students' responses could have a tremendous amount of flexibility in the way they were structured and worded. In this system, the student responses must be free of spelling or grammatical errors and formulated in a single sentence. The grading is based on bag-of-words coordinate matching and synonyms are retrieved with WordNet [51]. Coordinate matching simply encompasses that each word in the correct answer is compared to the corresponding word in the student's response. Initially, each word is checked for exact matches, which awards one point. If there is no match, the system will check for related words from WordNet and if there is a match there, it will award partial credit. No credit is awarded if there is no match.

3.2.3 Corpus-Based Methods

The Corpus-Based era introduced the usage of statistical properties of large document corpora [8]. Statistical properties are particularly useful when interpreting synonyms in short answers. One of the most known contributions in this field is the introduction of Latent Semantic Analysis (LSA), which is presented in Section 2.6.

Atenea

Atenea (Alfonseca et al. [1]) compares the accuracy of several variations of the BLEU algorithm when applied in automatic grading. The BLEU algorithm [33] originally encompassed evaluating the quality of text that has been machine translated from natural language to another. The central idea behind BLEU is that the closer a machine translation is to a professional human translation, the better it is. Therefore, quality is considered as the similarity between a machine's output and the translation performed by a professional translator. In regard to Atenea, the BLEU algorithm calculates an n-gram precision metric by finding the percentage of n-grams from the candidate translation that appears in any of the Reference Answers (answers written by teachers). Alfonseca et al. consider that the students' responses are the candidate translations, and the teacher writes a set of correct answers with a different word choice to be taken as a reference. The authors worked with a dataset that was made of seven questions and 885 Student Answers. All the answers were also scored manually by two different human judges. These judges also wrote the Reference Answers. In addition, Alfonseca et al. implemented a brevity penalty factor to ensure that translations that are missing information do not get higher results than complete translations that are not fully accurate. Furthermore, NLP was utilized as simple modifications to the original algorithm. Results from the project revealed that the correlation values between BLEU and the manual scores were better than those obtained with other keyword-based evaluation metrics used in existing systems.

Mohler
Mohler et al. [30] explore unsupervised techniques for automatically grading short answers. The main contribution here is that they individually compare eight knowledge-based measures of semantic similarity and two corpus-based measures of semantic similarity. These knowledge-based measures are shortest path, Leacock & Chodorow, Lesk, Wu & Palmer, Resnik, Lin, Jiang & Conrath, Hirst & St. Onge [30], while the two corpus-based measures are LSA and ESA (Explicit Semantic Analysis)[1]. When comparing the knowledge-based measures, Mohler et al. derive a text-to-text similarity metric by using a methodology proposed in Mihalcea et al. [28]. In simple terms, this technique utilizes each open-class word from one of the input texts to couple with each open-class word in the second input text to get the highest level of semantic similarity possible. More specifically, for each word W of part-of-speech class C in the instructor's answer, they find maxsim(W, C):

$$maxsim(W, C) = maxSIM_x(W, w_i) \tag{3.1}$$

where w_i is a word in the Student Answer of class C, and the SIM_x function is one of the knowledge-based measures mentioned above [30]. All these word-to-word similarity scores were also summarized and normalized with the length of the two input texts. Results from this methodology indicated that regarding the knowledge-based measures, the shortest path measure gave the best correlation. Corpus-based measurements are different from knowledge-based approaches in a way that they do not require any encoded knowledge of a text language's vocabulary or grammar. Mohler et al. ran a Singular Value Decomposition (SVD) operation on several corpora like the BNC (British national corpus) and the English Wikipedia. Results from the Corpus-based measures displayed that the LSA BNC measure provided the highest correlation. In addition to comparing these measures, Mohler and Mihalcea also considered incorporating the highest-scoring Student Answers with the teachers' answers to increase the vocabulary of the teachers' answers.

3.2.4 Machine Learning

Usually, a classification or regression model is used in machine learning systems to combine a variety of metrics taken from NLP and other sources into a single score/grade. A model can be built using student responses and the corresponding instructor grades, with the objective being to predict which score should be given to new answers [15]. An automatic short-response grading system can be created and made ready to use with a certain level of confidence by gathering data, creating the model, and then evaluating it. Decision trees and support vector machines are common learning algorithms in this area, in addition to using bag-of-words (Section 2.5.1) and n-grams (Section 2.5.2).

e-Examiner
e-Examiner is a tool developed by Gütl [19] to support the assessment process by automatically creating test items, assessing students' answers, and then providing feedback. Gütl has put three plugins into use for the automated evaluation task:

1. Using a vector space model, a COSIN Text Similarity plugin calculates the degree of similarity between a candidate response and one or more reference replies. This plugin allows e-Examiner to assess the similarity between the student's written response and the reference or model answers.

---

[1] Explicit Semantic Analysis (ESA) is a technique that represents the meaning of text in a high-dimensional space using a vector representation.

2. 'ROUGE measures plugin' (Recall-Oriented Understudy for Gisting Evaluation) [26] is a set of measures commonly used in natural language processing to evaluate the quality of automatic text summarization. By comparing the student's written response with the reference or model responses, the ROUGE Measures plugin in e-Examiner makes use of these metrics to evaluate the quality of the response. ROUGE measures calculate how similar the student's response and the reference responses are on the basis of how many n-grams (contiguous sequences of n words) they overlap.
3. The Assessment Score Builder plugin determines an overall score for a student's response that can be used for grading and feedback. It takes various factors into account, such as the similarity between the student's response and the Reference Answers measured by the COSIN Text Similarity plugin, the quality of the response evaluated using the ROUGE measures plugin, and potentially other criteria or rules defined in the e-Examiner system.

For each of the experiment setups, the parameters for the linear combination are calculated by applying linear regression on the output of the training dataset. The majority of the work encompasses the architecture of the system, which is focused around an open and flexible approach. A flexible design is essential for ensuring that the system can be utilized independently or quickly integrated with existing e-learning programs.

Hou
The work of Hou et al. [22] focuses on improving the interaction between students and teachers by implementing an intelligent computer system that is capable of automatically evaluating Student Answers. Hou et al. approach the assessment problem as a classification issue, classifying student results into two classes, such as 'good' and 'not good' in the eyes of the teacher. This is performed so that the teacher can quickly determine most students' learning situations for a particular subject, and then decide whether to continue teaching or stay to improve the material. Based on a collection of Student Answers, Hou et al. present a method for extracting related information and then creating a set of new features that represent each answer. This is executed in three parts:

1. Data preprocessing: removal of punctuation, decimal numbers and stemming.
2. Feature extraction: POS tagging, TF, TF-IDF, and entropy are utilized.
3. SVM classification: A Support Vector Machine (SVM) [16] classifier is applied to the assessment task by grading each answer from zero to ten. The classifier divides the responses into two categories: category +1 for those with six or more points and category -1 for those with fewer.

With this approach, a precision rate of 65.28% was achieved when all the features (POS tagging, TF, TF-IDF and entropy) were utilized. In comparison, the average precision rate was only 55.62% when only the TF and part-of-speech tag were used as features. This indicates that the use of TF-IDF and entropy as features have a positive influence on the precision rate.

3.2.5 Clustering

Later in the 21st century, the idea of grouping together similar responses was introduced. This can be performed by clustering Student Answers into groups to explore if student replies have any common characteristics. Such a method can reveal naturally occurring clusters of answers having similar structures to how students frequently respond. A majority of the work done under this theme encompasses identifying groups of related responses, which are then analyzed using both computerized and manual methods. One of the key benefits of this strategy is that it allows for a creation of a new system that enables teachers to provide common feedback to students belonging to the same cluster, and thus, the same characteristics. The teacher can accomplish this by choosing a prototype solution or answer from each cluster. Cluster analysis involves grouping objects in a way that makes them more similar to each other than to those in other clusters. Essentially, it is a collection of objects based on their similarity and dissimilarity. One of the most popular clustering methods is DBSCAN, which is a density-based algorithm. This encompasses that the algorithm looks at the number of points within a particular radius of the point itself. Another popular clustering algorithm is the k-means clustering algorithm which starts with a group of randomly selected centroids to act as entry points for each cluster. Subsequently, repetitive calculations are performed to optimize the position of the centroids. Typically, this is performed by calculating the mean of the points in the cluster from the last iteration of the algorithm.

Powergrading

Basu et al. [4] introduced an approach referred to as 'powergrading'. The goal of powergrading was to simplify the human effort for short-answer grading. This was attempted by grouping responses into clusters and sub-clusters, allowing teachers to grade multiple responses within a single action and provide detailed feedback to groups of similar answers. They also focused on finding consistent patterns of misunderstanding amongst students that went beyond the fraction of getting a question right or wrong. When the system automatically found groupings and sub-groupings of similar answers from a large set of answers to the same question, they found that answers for a particular question often clustered into groups around different modes of understanding and misunderstanding.

Horbach

One proposed approach presented by Horbach et al. [21] investigates the potential of answering clustering and label propagation as an approach to semi-automatic scoring. The focus here was finding the trade-off between grading accuracy and the reduction of teacher workload. In addition, their approach assumes that highly similar student responses are likely to receive the same grade from a teacher, and thus, can be grouped and graded as a single unit. To achieve this, clustering techniques based on surface features were used. Answers were clustered automatically, and the teachers would then label/grade only one item per cluster. That label would then be propagated to the other items in the same cluster. 1668 short answers to 21 questions were utilized together with sample solutions provided by teachers for each question, as well as teacher-assigned grades for each student response. This data came from a listening comprehension task for students learning the German language. They extracted features based on word n-grams, character n-grams, and keywords. For each item, cosine similarity between its feature vector and the centroid of each cluster was calculated. Furthermore, Horbach Et al. evaluated three methods for labeling the questions but concluded that the selection and grading of the closest item to the centroid of a cluster and then propagating this label to all other members of the cluster led to the highest accuracy. With this approach, they achieved a reduction in teachers' workload to labeling only 40% of all different answers for a question, while still maintaining a grading accuracy of more than 85%.

3.3 Most Relevant Approaches

CAA systems can vary greatly in the way they evaluate text answers. Consequently, there exist several different approaches to automatic grading and feedback retrieval. This section will cover three present and more relevant existing approaches to automatic grading and feedback retrieval in the context of this thesis.

3.3.1 Automatic Evaluation of Short Free-Text Answers

Multiple approaches to evaluate short free-text answers have been performed. An approach and implementation for automatic assessment of short free-text answers are proposed by Rodrigues, Fatima et al. [38]. This approach is based on Reference Answers (RA) and Student Answers (SA) and is composed of four modules:

- A classification module that gives a teacher the possibility to classify each question and answer into types. This module is used to have the teacher set requirements based on the question/answer type, to increase overall evaluation performance.
- A preprocessing module, consisting of multiple steps to process the text for further evaluation. The preprocessing steps taken here are similar to the steps described in Section 2.4.
- An evaluation module that evaluates student responses. This evaluation is based on correct RAs, typically answered by teachers. The Vector Space Model (VSM) is used to apply term weights, and each SA and RA is represented by vectors with the corresponding term weights. The similarity between the teacher/Student Answer pairs is calculated using cosine similarity.
- A feedback module that provides comments to students based on the missing topics in the SA. The feedback covers the student's performance and errors, as well as hints for improvements. Feedback is also given to the teacher on topics that are less covered by a group of students or individuals. A word-matching algorithm is used to provide the feedback, where the canonical form of RAs is compared with the canonical form of the SAs to look for similar words. Each word from the SA is searched for in the RA, and if it is not found, the part-of-speech (POS) tag for the word is searched for in the RA. If the tag is found, the word is searched for in a list of synonyms associated with the tag in the RA. This is done as an effort to better match the RA and SAs, and since both answers are in canonical form and it is performed on short free-text answers, each word should be meaningful in the context of evaluation.

3.3.2 Automatic Short-Answer Grading And Feedback Using Clustering Methods

The work conducted by Suzen et al. [41] presents a concept of automatic grading of short-answer questions by employing a clustering approach. Standard data mining techniques are applied on a corpus of SAs and similarity is calculated with RAs. The similarity is based on the number of common words. Their text representation is based on a bag-of-words approach, which is described in Section 2.5.1. Here, each SA is represented as a bag-of-words, where the idea is to extract unique words from the collection of documents. Subsequently, the documents are vectorized and similarity is calculated using Euclidian distance. K-means clustering is applied to cluster the documents into k clusters, and the documents are assigned based on the distance to each cluster. This way, the documents that are similar, or have a short distance to each other get clustered together and assigned the same grade. Similar feedback is also provided to each answer within a cluster. Three clusters named

Excellent, *Mixed* and *Weak* were utilized. All documents in the same cluster were expected similar in terminology and term usage, and the words in each cluster were compared to demonstrate that the clusters were formed based on the number and selection of words from the RA that have been utilized.

All SAs were graded similarly by two graders, with a few inconsistencies. A high correlation was calculated for the grades given by the graders and the distances between documents. This entails that similarly graded SAs would have a low distance to each other and therefore be clustered together. In addition to the clustering method, the work also showcases a description of how SAs correlate to the grades based on words used. A basic calculation is used to calculate the number of similar words used in a SA and a RA. In this case, Hamming distance was used. When the distance was 0, all words in the RA were used in the SA. Note that this was performed after the preprocessing process and that usage of a synonym list was not used, but the idea was clearly presented. Further on, the use of Hamming distance to predict grades (or marks) in SAs was presented, including a hypothesis that involves whether the number of similar words in a SA to a RA can be used to predict grades. To sum it up, due to the high correlations between grades graded by graders, word similarity, and clustering presented in this work, this idea for predicting grades and feedback in short-answer texts can with high confidence be implemented for other scenarios and should be tested further.

3.3.3 Automatic Evaluation of Students' Answers with LSA

Kanejiya et al. [23] present an approach for automatic evaluation of Student Answers using syntactically enhanced LSA (SLSA). LSA is mainly a corpus, bag-of-words based NLP technique that is used to find semantic similarities between texts. Typically, a Term-Document Matrix is constructed and preprocessing steps are performed. From there, Singular Value Decomposition (SVD) is applied to reduce the dimensionality of the matrix, and cosine similarity is used to calculate the similarity between documents, or SAs and RAs. Kanejiya et al. argue that an obvious way of improving the performance of LSA is to incorporate syntactic information into it. Using a part-of-speech (POS) tag is such an incorporation, and a proposed approach for such an implementation is given by using the preceding POS tag of a given word in the matrix. The matrix would then contain rows with word-POS tag combinations and columns would be documents. An argument is given that by using the preceding POS tag, a syntactic neighborhood is made around the focus word, allowing two documents to be compared based on their syntactic-semantic regularity and not only semantics.

This approach initializes by first running a POS-tagger to turn the corpus into a POS-tagged corpus and then creating a matrix of word-tag pairs, where the tag is the tag of the preceding word. The similarity between documents could now be calculated within this vector space by applying cosine similarity. To predict the performance of this approach, correlations between this technique and human evaluators were performed, as well as regular LSA techniques. The work concludes that this innovative approach has slightly, limited improvements in relation to LSA, but leads to future experiments in the syntactic neighborhood of terms and might be worth investigating further.

Chapter 4

Design

Two approaches will be presented for automatic grading and feedback. This chapter first presents the high-level goal of this thesis. In addition, it covers a high-level design of the two approaches and potential use cases for them both, whereas the descriptive implementation and findings are covered in Chapter 6. The first approach features a Reference Answer (RA)-based approach that will be compared to Student Answers (SA) by means of a similarity calculation. The second features a clustering approach, and is based on the k-means algorithm that optimally clusters together answers of similar grades and provides feedback to whole clusters.

4.1 High-level Goal

The high-level goal of this work is to provide an initial blueprint for a comprehensive system that can be employed for automatic grading and feedback generation to students. The system is made with the goal to relieve work from teachers that spend considerable amounts of time grading and giving feedback, as well as providing feedback to students to improve learning. Furthermore, the biases in having a teacher evaluate answers will optimally be removed. Such a system can also provide students with immediate feedback, enabling them to see where they can improve and take appropriate steps to do so right away. Moreover, the feedback can also be used to help explain the grade the particular student has been given. Automatic grading and feedback systems can also be scalable, making them capable of grading large numbers of responses quickly and accurately. In a full-scale solution, the data retrieved from the grading process can be utilized to generate valuable insights into students' performance and learning outcomes, which a teacher can use to improve the learning effectiveness and experience for the students. The two approaches presented in this thesis are intended to serve as a step in the process to develop such a system as well as test out and explore various techniques that can be incorporated to improve grading and feedback performance. Figure 4.1 presents an overview of such a system.

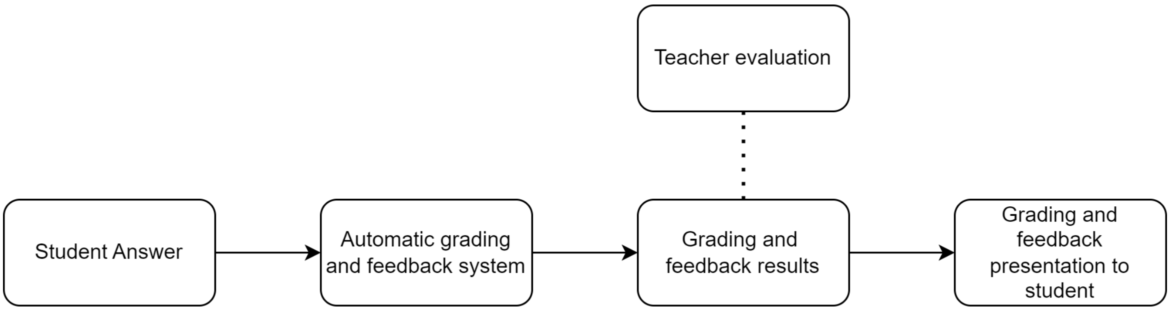

Teacher evaluation
Student Answer
Automatic grading and feedback system
Grading and feedback results
Grading and feedback presentation to student

Figure 4.1: High-level goal

The system would work by having a student's short answer inputted into a system for automatic grading and feedback generation. Instead of having the teacher evaluate all answers individually, the system would aid in this process and the teacher would function as a quality assurance link. This way, the teacher would spare time while increasing the grading and feedback quality due to limiting the biases from manual grading. The main focus in this thesis, however, is on how an automatic grading and feedback system can be implemented, and less on the teacher evaluation part.

Feedback

The feedback technique that will be addressed in this thesis encompasses, as explained in Section 2.1 the utilization of providing keywords to students. These keywords represent important concepts or themes that are absent in a particular student response and could be beneficial feedback terms for the student. To retrieve these keywords, the Student Answers will be compared to a set of terms that are known, or is likely to be of high quality for that particular question and should be present in a high-quality response. Figure 4.2 illustrates a simplified version of the feedback/keyword retrieval technique.

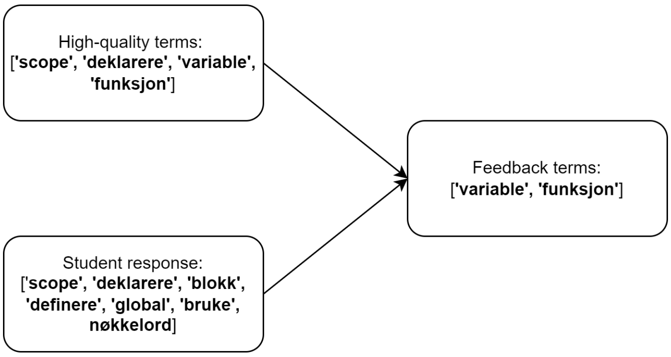

High-quality terms:
['scope', 'deklarere', 'variable', 'funksjon']
Feedback terms:
['variable', 'funksjon']
Student response:
['scope', 'deklarere', 'blokk', 'definere', 'global', 'bruke', nøkkelord]

Figure 4.2: High-level feedback technique

A set of already-known feedback terms are present, and these terms are looked for in a particular student response. If the term is absent in the student response, it is a possible feedback term. This idea for feedback generation also requires a type of evaluation method to measure the effectiveness of the technique. Thus, methods for feedback evaluation will also be discussed throughout this thesis.

4.2 Comparing Student Answers with Reference Answers

Since it has been found that the use of LSA-based approaches can perform as well as an intermediate expert human evaluator to evaluate text and that such approaches are rather popular as well, it is given that it would be interesting to test out such an approach in this case. However, it has not been proven that LSA can perform as well as an accomplished expert of the domain [23]. On this basis, at least assessing a similarity-based approach on the Norwegian datasets is highly relevant, and optimally attempt to improve the performance found in earlier approaches. Consequently, this approach features a combination of the work described in Section 3.3.1 and 3.3.3. Figure 4.3 showcases a high-level representation of how such an approach will look like.

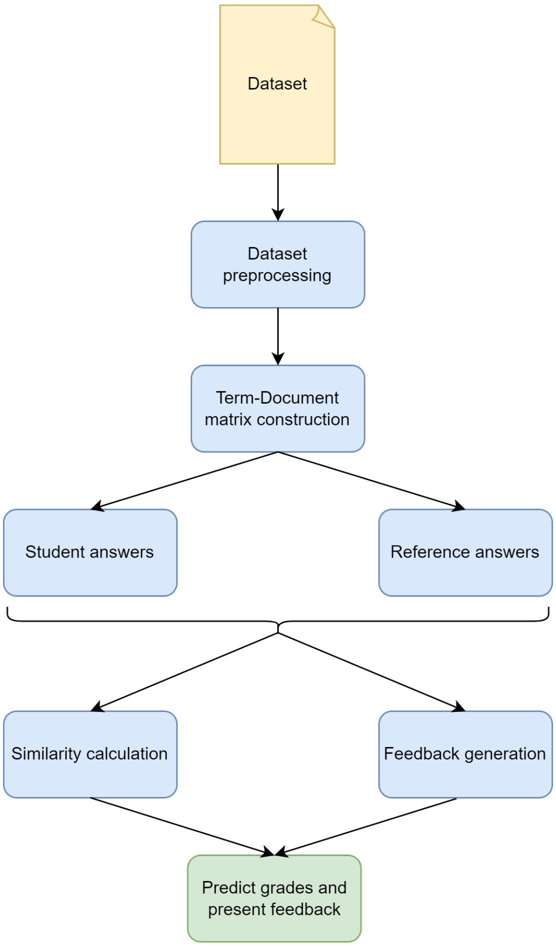


Figure 4.3: High-level similarity approach

The approach is based on finding similarities between RAs and SAs. Firstly, preprocessing techniques will be performed on a dataset and a Term-Document Matrix will be constructed. From this construction, Reference Answers and Student Answers will be retrieved. Since the datasets currently only consist of SAs and no RAs, it would be an idea to either create model answers for each question featured in the datasets, or use some of the highly graded SAs, and have them act as RAs for this approach. This is a process that would have to be done manually with care to pick/develop a good enough model answer. Furthermore, the best solution here would have to be tested out on the datasets. Another relevant proposal is using only the most common words across the datasets, and having them act as a RA. The similarity measure would in this case be to calculate the SAs against the most common words. These words can be handpicked or found by extracting the words with the highest weights from the highest-graded answers (the most relevant in the datasets) above a given threshold, and then computing the cosine similarity to RAs.

LSA is described in Section 2.6, and for this approach, the datasets consisting of all SAs will act as our corpus. Appropriate preprocessing techniques described in Section 2.4 will be performed and the resulting data will be put into a bag-of-words manner. From there, a Term-Document Matrix will be constructed, where SAs will act as individual documents in columns and terms will be rows. Each individual cell entry in the matrix represents a term weight of term i in document j. Generally, TF-IDF weights will be used to weight the terms, but other weighting techniques will be explored. Singular Value Decomposition is attempted to only retain the largest values (weights) in the matrix. From there, cosine similarity will be used to calculate similarities between SA and RAs. If an approach of high performance results in multiple RAs for each question, similarities can be calculated for each RA, and some type of weighted average can be performed on the similarity measure against all RAs.

As for the feedback generation, a highly relevant approach is proposed by the work described in Section 3.3.1. Such an approach can appropriately be implemented for our approach as well. Once the SAs and RAs have been vectorized, word similarities can be identified between the two. For each word in the RA(s), it can be looked for in the SAs. If the word does not exist, it is a potential word to be used for feedback for that particular answer, as it may be a terminology word that identifies an unanswered theme. However, if the word does exist, it would mean the SA contains a correct source of information. Implementing a synonym list is also relevant for the feedback technique, as the RA and SA may be using different words that mean the same. Hence, the same meaning can be achieved with different terminology and must be identified to provide meaningful feedback. This way, specific feedback is generated for each SA. It is important to note, however, that several preprocessing steps have been performed along with Singular Value Decomposition, so the idea is that all words that are left in the Term-Document Matrix are important or relevant. Hence, all words that are put into the word-matching/feedback algorithm are of meaning and could be used as relevant feedback.

The feedback techniques that are implemented in this approach rely heavily on the performance achieved by the grading algorithm. Using common words in the dataset, or multiple RAs would result in and affect the feedback technique. Thus, a quality evaluation must be performed to identify the particular feedback technique that performs the best. As an example, if the usage of common words in the datasets provides high quality for grading, the same common words might be useful for feedback generation. In such a case, the word matching would occur against SAs and the common words.

4.3 Clustering

Since grouping similar responses has grown in popularity during the 21st century, one of the proposed approaches in this thesis will investigate possibilities regarding clustering. Related work in this field of research can be read more about in Section 3.2.5. For this approach, clusters of similar answers will be constructed based on the content of the SAs. One of the objectives here is to examine how answers of various grades are represented in different clusters, meaning whether answers of similar grades are grouped together or if the clusters comprise answers of all grades. The possibility to create a feedback system that enables teachers to provide similar feedback to students in the same clusters will also be examined. As future work, Basu et al. [4] provide a clear direction forward in the clustering field of research. The authors state that there are still many questions regarding how these capabilities can be best presented to a human operator, including how clusters and sub-clusters are displayed and how feedback can be provided to students.

Sections 3.2 and 3.3 present three existing approaches in the clustering area. Suzen et al. [41] introduce an approach that depends on the similarity between the SAs and RAs. Since the datasets do not include Reference Answers, the clustering approach will focus on a similar approach to the two works done by Basu et al. [4] and Horbach et al. [21]. These two papers focus on grouping responses into clusters without the use of model answers (RAs). As a result, teachers are able to grade multiple responses with a single action as well as provide detailed feedback to groups of similar responses. The approach will also utilize the concepts of NLP (Section 2.4) to preprocess the data before initiating the clustering process by employing the common k-means clustering algorithm.

A high-level illustration of this approach is presented in Figure 4.4. A dataset is first sent into the NLP preparation pipeline where it is used to create a Term-Document Matrix for each question. The k-means clustering algorithm will attempt to cluster together similar responses, assign grades, and produce feedback for these groups of responses. This system is regarded as a semi-automated grading system as it allows for teacher manipulation. A teacher can optimally enter the system and display the groups of responses in a clear way. Then, the teacher can step in and modify the common grading and feedback suggested by the system. After the teacher has approved the grading and feedback given by the system or modified it, the grading and feedback may be displayed to the students.

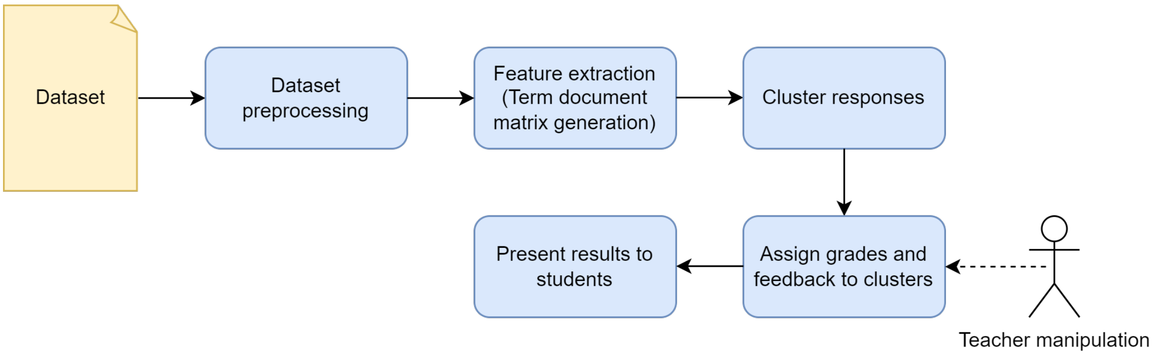

Dataset
Dataset preprocessing
Feature extraction (Term document matrix generation)
Cluster responses
Assign grades and feedback to clusters
Present results to students
Teacher manipulation

Figure 4.4: High-level clustering approach

K-means clustering is a well-known unsupervised learning algorithm that is used to divide a dataset into k clusters. Here, each data point (student responses) is assigned to a cluster such that the distance between the data point and the centroid of the cluster is minimized. The method seeks to group together data points that are similar to each other while at the same time keeping data points in different clusters as dissimilar as possible. The k-means algorithm is a well-known clustering technique in many fields, including computer science, engineering and social sciences. It can be applied in a variety of domains, including Information Retrieval, image processing, and Text Mining.

The k-means algorithm can be broken down into four main steps:

1. Randomly initialize k cluster centroids: These initial centroids are selected by randomly selecting k data points from the dataset.
2. Allocate data points to clusters: Measure the distance between each centroid and each data point in the dataset. This is usually done by utilizing the Euclidian distance metric. Assign each data point to the cluster whose centroid is nearest.
3. Update the centroids: Set the new centroid of each cluster to be the mean of the data points in each cluster.
4. Repeat step 2 and 3 until convergence: Re-assign all data points to the clusters based on the updated centroids and recalculate the centroids. Quit when the centroids and cluster assignments converge to stable values.

Although being a strong and popular clustering technique, the k-means algorithm has a few limitations. The algorithm assumes that the clusters are spherical and equal in size which may not be true for all datasets. Another drawback is that the user must choose the number of clusters k, which is a hyperparameter[1]. This hyperparameter might have an impact on the clustering performance. K-means clustering is nevertheless a useful and popular method for grouping data.

---

[1] A hyperparameter is a parameter that is chosen before a learning algorithm is trained and is used to control the learning process

Chapter 5

Dataset

The datasets utilized to explore different approaches originate from two exams in the course IT2810 - Web Development. This course is currently being taught at the Norwegian University of Science and Technology (NTNU) and the exams are from the years 2018 and 2019. The course covers technology and methods used for web development, including frameworks, architecture, languages, and formats exercised for standard web applications. For both exams, the same professor was responsible for grading, meaning that all answers received one grade from the same sensor ranging from 0 to 5, where 0 is the lowest and 5 is the highest. Furthermore, all answers are mostly answered in Norwegian, either in Norwegian Bokmål or Norwegian Nynorsk, but a subset were also answered in English. The whole dataset consists of short answers, however, the answer length for each answer, or the average answer length for one exam might differ greatly and should be taken into consideration.

The 2018 dataset (D18) has 16 questions and 164 students participated, whereas the 2019 (D19) dataset has 8 questions and 185 participants. An overview of the questions can be found in Appendix B. These exam questions were meant to test the knowledge of the students throughout one whole semester of learning in the course. Most questions and answers are written in natural language. However, some answers contain code examples in addition to text, and a few questions specifically asked to use examples. Certain questions were also pure programming questions, and because this work focuses on automatic grading and feedback generation of short answers, it might be an idea to exclude such questions. Performing evaluation of code, as well as short-answer text would fall outside the scope of this thesis.

The grading process operates in a way that all question responses were given a grade ranging from 0 to 5. These grades were not given directly to the students. Instead, they were given an overall grade calculated from all the question grades. Each question in the datasets had a weight assigned to them indicating the importance of the question, and the overall grade was calculated by adding together the sub-grades multiplied by the weight. In these two datasets, however, all question weights were set to 1.

The datasets contain some noise. Not every student has answered every question, and some answers contain unrelated answers. Such answers are common on exams and an argument can be made that a realistic grading- and feedback system should manage such noise. Hence, these answers should probably stay in the dataset and not be excluded.

Although this course and the datasets are mostly written in Norwegian, the course content contains a high degree of English terminology which is included in all answers. Such words are important because they are typical for the topic, and should be handled in the system. Potentially, the lack of these words in Student Answers might be a good indication of the student's knowledge level and could function as meaningful feedback for the student.

The two exams were taken in Inspera, which is a digital assessment platform utilized by NTNU. Consequently, the datasets include some unnecessary attributes that should be cleaned before applying IR techniques. These attributes include general Inspera data that is irrelevant to this project, as well as candidate IDs for all the students and the duration spent answering each question. The most relevant attributes for this project are the Student Answers along with their final course grades and a set of attributes that uniquely identify each exam question. In addition to this, all attributes are written as nested JSON objects. These objects should first and foremost be normalized and the relevant attributes should be selected. Another notable element in the datasets is that the answers were stored with HTML tags from Inspera. Optimally, only pure text written by students is valuable to investigate, and it is therefore uncertain how a potential system would react if HTML tags are kept. On this basis, removing them from all text answers would be beneficial.

An important aspect of analyzing such datasets is looking at the grade distribution. Based on the distribution, it can be determined to some degree which techniques might not perform well. If the grade distribution of the dataset is unbalanced, it can affect the performance or provide implications related to automatic assessment. An example of this could be within a machine learning approach where you have a test and training set. The training set is used to train or learn a pattern to predict grades in the test set, and consequently, an unbalanced dataset could provide biases toward grading. Hence, an analysis of the grade distribution in the two datasets is performed in Section 5.2.1

5.1 Data Cleaning

During the data-cleaning process, a plan was constructed to keep track of what needed to be done with the datasets. This plan included a number of problems/obstacles that were discovered during the data exploration phase. Table 5.1 lists these obstacles.

Table 5.1: List of obstacles in the datasets

Id	**Obstacle**
1	Inspera formatted text
2	Presence of programming questions
3	Presence of programming snippets in responses
4	Responses answered in different languages
5	Presence of tables and images in responses
6	Special characters
7	Responses not answered
8	Typos in responses
9	English terminology in Norwegian responses
10	Size of datasets
11	Grade distribution

The original datasets were, as mentioned, written in a nested JSON format and include 33 properties that contain metadata about the exam, candidates, responses, and other essential exam data. Therefore, the first step in the data-cleaning process was to extract the relevant attributes for the use case of this thesis and construct a new processed JSON file only containing the attributes needed. These attributes included IDs to uniquely identify candidates and questions for both exams. The Manualscores (the grade given by a teacher on each SA) and complete SAs were also retrieved. The new, processed JSON file was then saved with the HTML encoding 'utf-8', causing the Norwegian letters 'ÆØÅ' to be displayed properly in the file.

Every exam was conducted online and completed utilizing the Inspera platform. On this platform, students are given the option to format their responses. The students can incorporate bold text, lists, equations and titles in their responses. Consequently, the responses were noted with various HTML and markup language tags that described the metadata of the text. This challenge is listed as Challenge 1 in Table 5.1, and could be handled with a regex function that removed such markup language. However, given that these exams cover the subject of web development, several candidates also included HTML elements as part of their solutions. Ergo, deleting the HTML tags would also delete portions of the student responses. The solution here became removing all markup and HTML from the text, including HTML constructed by the students. This can be justified by the fact that the text was overflowing with Inspera-generated HTML and markup, and because it does not exist any openly available overview over which HTML tags Inspera uses, causing filtering them out to prove problematic.

The students had the option to add tables and images to their responses. These were also marked with HTML tags. Because of the way HTML structures tables, deleting the tags and keeping the text would leave a text structure without a way of identifying which cell content goes to which column. In addition, only seven responses from D18 and nine answers in D19 contained tables. Hence, the datasets would not be significantly affected by removing them. Therefore, the decision to remove the tables with the same regex function utilized to remove the other Inspera markup language was made. The regex function applied was manually constructed for this specific task.

One answer in D19 and eleven answers in D18 included images. It was determined that removing all responses that included image tags would be the simplest approach to take. This was achieved by detecting if the answer had an HTML image tag and removing these responses from the datasets. It is important to note that removing only the images from these replies were discussed, but that would remove only a part of these responses and could cause the grading to be inaccurate. Furthermore, keeping the images in the datasets would encompass evaluating them as well, which would make the proposed approaches substantially more complex, and is outside the scope of this thesis.

Challenge 2, the presence of programming questions was also taken care of in the data-cleaning process. As question 8 in D18 was a pure programming question, asking students to write the code for a React component that included an H1-element with the text 'Hello World', this question was dropped in the processed JSON file. Such questions are not covered in this thesis as it focuses on natural language short-answer responses in Norwegian. An example of the processed JSON file with one candidate and one response is presented in listing 5.1.

```
{
"candidates": [
  {
    "candidateId": "10090",
    "questions": [
       {
         "questionId": "38102438",
         "questiontitle": "Web storage",
         "maxQuestionScore": "1",
         "Manualscore": "5",
         "response": "At en har mulighet til å lagre state asynkront
         lokalt på maskinen/mobilen."
        },
        ...
```

Listing 5.1: Snippet from processed JSON file

A subset of the students chose to not answer some questions, which is referred to as Challenge 7 in Table 5.1. Such unanswered responses would contain a string with the text ‘None’. In addition, a portion of students chose to write a simple explanation about why they could not answer that question. Some examples of this are responses such as ‘Denne var tricky’ which can be translated to ‘This one was tricky’, and ‘Beklager men dette er et spørsmål jeg ikke kan svare uten å google.’, meaning ‘Sorry but this is a question I can’t answer without googling.’ in English.

The inclusion or exclusion of these responses in the datasets had to be decided. To achieve a more realistic grading result, the decision to keep these responses in the datasets was made. Consequently, a set of responses that received a score of zero were kept. As a result, this caused the datasets to represent a more realistic response collection.

The next step in the data-cleaning process involved normalizing the data and creating a dataframe. The Python library ‘pandas’ was utilized for this process. This is a library specifically made for data analysis [32]. Such a dataframe is arranged in rows and columns as a two-dimensional data structure, allowing for easy manipulation of the data. The normalization process was utilized using the ‘json_normalize’ function from pandas[1], and functions by taking a JSON structure as input and converts it into a two-dimensional dataframe.

Another issue involved handling the responses in different languages, referred to as Challenge 4 in Table 5.1. Although Norwegian Bokmål was used for the majority of the replies, English and Norwegian Nynorsk were also utilized in some responses. As a result, words with the same meaning would not be immediately regarded equal in a straightforward vector similarity technique. As this thesis prioritizes short-answer grading and feedback generation in the Norwegian language, a decision to singularly keep the Norwegian Bokmål answers and remove those in English and Norwegian Nynorsk was made. Additionally, due to the exam’s subject ‘Web development’, several questions answered in Norwegian Bokmål also contained English terminology (Challenge 9 in Table 5.1), causing a potential multi-lingual dataset issue. These terminology words include words like ‘grid’, ‘arrow’, ‘props’, ‘layout’ and ‘state’. A Python library called Polyglot was utilized to address the language issue.

---

[1] `https://pandas.pydata.org/docs/reference/api/pandas.json_normalize.html`

The Polyglot library [48] is an NLP toolkit for Python. It provides support for various tasks such as language detection, part-of-speech tagging, named entity recognition and sentiment analysis. The library utilizes various language models and algorithms to perform these tasks, making it an all-in-one solution for a wide range of NLP requirements. Polyglot's language detector can detect both English, Norwegian Bokmål and Norwegian Nynorsk, making it well suited for the language issue. The language detector can identify which language appears most frequently in the text, as well as the confidence level for each response. This level is an approximation of how confident the detector is of the suggested language.

The language detector was run on each response. Subsequently, the most probable language and the degree of confidence were added to their own new columns in the dataframe. The resulting dataframe showed that the detector was not entirely accurate. One of the main issues was that many of the Norwegian responses contained English terminology which resulted in a subset of responses being marked as English, but were in fact Norwegian bokmål. To effectively resolve this issue, it was decided to only keep all the Norwegian answers and the English answers that had a confidence level lower than 85. The idea behind this was that English responses with a confidence level lower than 85 were with a high probability Norwegian Bokmål answers, containing some English terminology. Conversely, English responses with a confidence level higher than 85 were most likely English responses and were therefore removed.

With this approach, the English responses were removed and most Norwegian responses were kept in the datasets. This implies that the method may not always be 100% accurate, but ought to ensure that the majority of the responses were properly classified. The number of replies that were identified in each language for each dataset is presented in Tables 5.2 and 5.3. Confidence levels over 85 are marked as 'High confidence', and under 85 as 'Low confidence'.

Table 5.2: Languages in responses data D18

Language	**Answers**
English	129
High Confidence (English)	67
Low Confidence (English)	62
Norwegian Nynorsk	84
Norwegian	2242
Responses kept	**2304**

Table 5.3: Languages in responses data D19

Language	**Answers**
English	111
High Confidence (English)	58
Low Confidence (English)	53
Norwegian Nynorsk	12
Norwegian	1357
Responses kept	**1410**

The size of the datasets is to be regarded as somewhat low, stated as Challenge 10 in Table 5.1. This is an obstacle that is not unusual in the field of short-answer grading, and is one of the core constraints of it [6]. The lack of data can cause the diversity of student grades to be relatively small, which can lower performance or cause approaches to perform inaccurately. Furthermore, selecting terminology words in a small dataset may prove difficult because there may be an imbalance in the datasets, causing a disproportionate number of answers of one particular type. Consequently, the choice of terminology words for feedback generation may be biased.

5.2 Feature Analysis

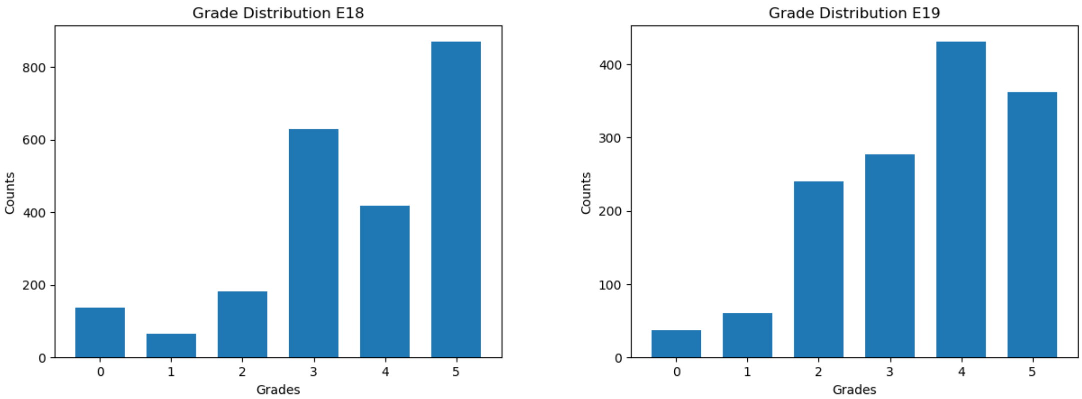
After the data-cleaning process, a feature analysis of the two datasets was performed to get a deeper insight into the two datasets. Feature analysis is important to perform to fully understand the data, and spot patterns, trends and outliers. Such an analysis can lead to spotting any biases and limitations. Furthermore, it can aid in making the right decisions for further data processing and modeling. Table 5.4 provides an overview of the most important features in the two datasets after the data-cleaning process was conducted. All grades and word count calculations were performed after the preprocessing step involving the removal of stopwords and only on the Norwegian Bokmål answers. The stopword removal process is elaborated on in Section 6.2.2.

Table 5.4: Feature analysis overview

Feature	**Exam 2019**	**Exam 2018**
Students participating	182	162
Number of Questions	8	15
Mean Grade	3.48	3.62
Median Grade	4	4
Standard deviation Grade	1.3	1.4
Answers graded 0	38	137
Answers graded 1	61	65
Answers graded 2	240	183
Answers graded 3	277	630
Answers graded 4	431	418
Answers graded 5	362	870
Mean Word Count	61	37
Median Word Count	52	31
Standard deviation Word Count	39	25
Min Word Count	1	1
Max Word Count	293	218

As depicted in Table 5.4, the mean grades are almost identical for the two datasets. The word count standard deviation for both datasets is also quite high, suggesting that the word counts in the responses are relatively spread out from the mean. A standard deviation of 1.3 and 1.4 for the grade indicates that the majority of the data points are within 1.3 units of the mean (above or below), however, some may be further away. The mean word count is also a bit higher in the D19 dataset compared to D18. This may connect with the fact that there are more questions in the 2018 exam, while both exams were given the same amount of time to be completed. The D18 exam also stated in the introduction that 'Short and precise answers are rewarded in the assessment'. The standard deviation for the word counts for both datasets is also high, indicating that the answer lengths are also relatively spread out widely from the mean.

5.2.1 Grade Distribution



(a) Grade distribution D18

(b) Grade distribution D19

Figure 5.1: Grade distributions

Figure 5.1 contains two plots of the grade distributions in the two datasets. Based on the plots, it can be seen that both datasets are not normally distributed. With most of the data centered around the mean and few data points at the extremes, a normal distribution has a distinctive bell shape. However, these plots illustrate that higher grades are more common than lower grades in the datasets. The skewed distributions are summarized in Table 5.5.

Table 5.5: Skewness and Kurtosis of datasets

	D18	**D19**
Kurtosis	-1.2	-1.41
Skewness	-0.79	-1.19

Skewness is a metric that represents the asymmetry of a probability distribution. A distribution has zero skewness if it is symmetric. Positive skewness means the distribution is skewed to the right, and negative skewness implies it is skewed to the left [17]. Both datasets have negative skewness, indicating that the tail of the distribution is longer on the left side, and the bulk of the data is located towards the right side of the distribution.

The measure of Kurtosis refers to the tailedness of a distribution [17]. The tailedness reflects the frequency of the outliers. The kurtosis in both datasets is also negative, meaning that the distribution is platykurtic. In comparison to a normal distribution, a platykurtic kurtosis is flatter and has thinner tails. This implies that, compared to a normal distribution, the data contains more values in the center of the distribution and fewer extreme values.

In the D18 dataset, the most common grade is 5, while in the D19 dataset, the most common grade is 4. The occurrence of 0 and 1 graded answers are low in both datasets. Handling this unbalance is essential to provide accurate and unbiased grading. In this case, the model might not generalize effectively and be biased towards the dominant classes, which are answers graded 4 and 5. This would cause an issue in a supervised machine-learning approach because the model is given access to the class labels in the training set and attempts to learn from that. If this training set contains mostly high grades, the model may become biased. However, because the k-means clustering algorithm is an unsupervised clustering approach, the datasets are not required to be normally distributed. Unsupervised clustering is a type of exploratory data analysis that seeks to group similar data points together only based on their features. K-means clustering is only based on the distance between data points and therefore can be used with both normally and non-normally distributed data. In addition, the unbalanced datasets are not a major problem in the RA-based approach either. This is because the methods utilized can handle both normally and non-normally distributed data since it makes no assumptions about the data's distribution.

5.2.2 Answer Length

The feature of answer length is also an important consideration in an automatic short-answer grading system. The length of the responses can be an indicator of the degree of detail the student offered in their answer and may also affect the grade that is given. However, especially in the D18 dataset, the instructor favored short and precise answers, entailing that a longer answer should hypothetically not give a higher score. It is nevertheless important to look at how the grades correlate with answer lengths. Table 5.6 and 5.7 provide an overview of the answer lengths for all grades in the two datasets after the data-cleaning process was performed. The 'answered' row represents the number of answers for each grade in the datasets.

Table 5.6: Word Count Statistics by Grade D18

	Grade 0	**Grade 1**	**Grade 2**	**Grade 3**	**Grade 4**	**Grade 5**
Answered	137	65	183	630	418	870
Mean	14.85	18.35	25.54	35.75	39.91	43.54
Median	13.00	15.00	23.00	30.00	33.50	36.00
Min	1	2	4	3	3	4
Max	53	74	76	121	170	218
Std	11.74	14.01	13.31	20.65	24.16	28.62

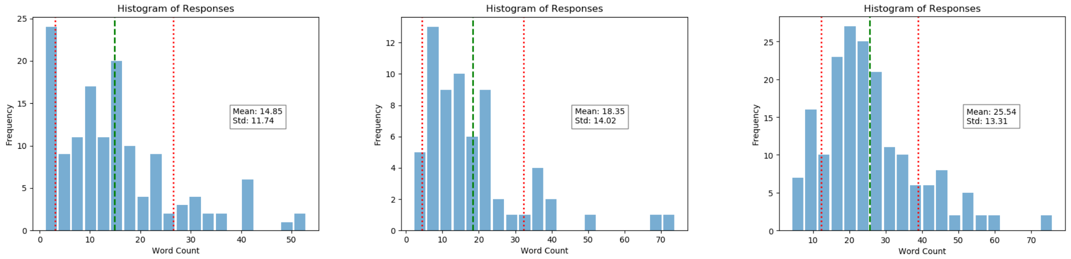
Table 5.7: Word Count Statistics by Grade D19

	Grade 0	**Grade 1**	**Grade 2**	**Grade 3**	**Grade 4**	**Grade 5**
Answered	38	61	240	277	431	362
Mean	13.68	29.74	45.39	68.60	63.91	70.49
Median	5.0	17.00	36.00	56.00	58.00	62.0
Min	1	4	9	11	10	13
Max	65	141	190	293	238	262
Std	16.89	32.54	29.70	42.57	35.80	38.93

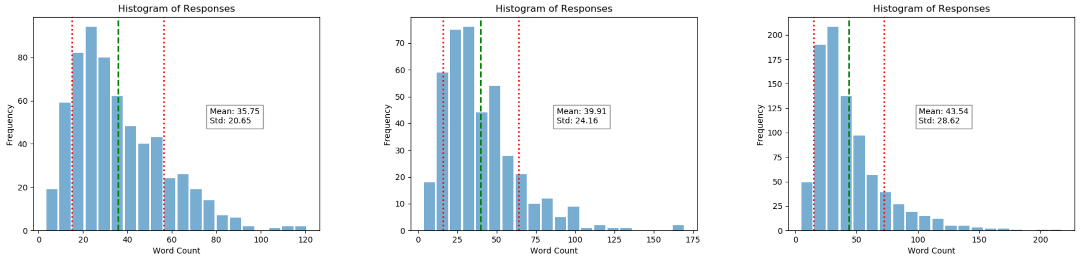
It can be observed that the mean and median word counts increase gradually for all grades in the D18 dataset. This can indicate that word counts positively correlates with grades. The minimum word count is generally the same for all grades in the D18 dataset, and the maximum word count increases gradually for each grade in the D18 dataset. Again, this suggests that the word count correlates with the given grade and can be incorporated as a feature in the grading process. However, the minimum word counts in the D18 dataset for grades 4 and 5 are low and can imply that answers of low word count can receive high grades as well.

For the D19 dataset, the mean word counts increase for all grades except from grade 3 to 4. The minimum word count is approximately the same for the grades in intervals 0 to 1 and 2 to 5. However, grade 5 has the highest minimum word count, indicating that there are no significant answers with a low word count that were graded 5. The maximum word count is also the highest for grade 3, suggesting that a high word count does not automatically give the highest score.

An argument can be made that standard deviations of word count per grade can reveal within what range of word count the majority of answers for each grade are in. This can be utilized to observe which grade range a new response is within and to further use that as a part of the grade prediction. By scaling up the correlation score depending on word counts and corresponding standard deviations, a more accurate grading calculation may be achieved. A smaller standard deviation suggests that answers of the same grade have similar word counts. Conversely, a greater standard deviation indicates varying word counts for responses of the same grade.



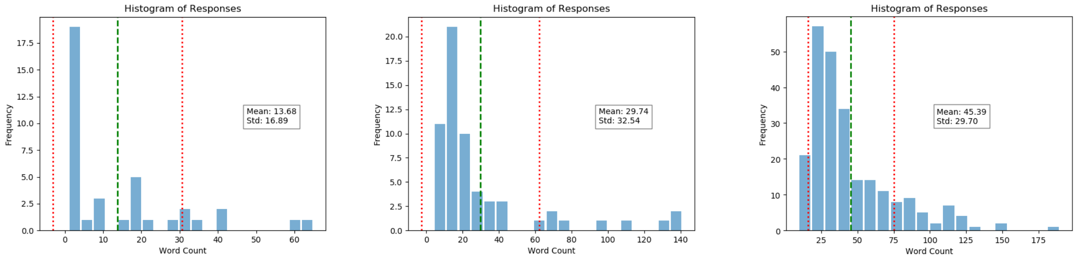
(a) Grade 0 (b) Grade 1 (c) Grade 2



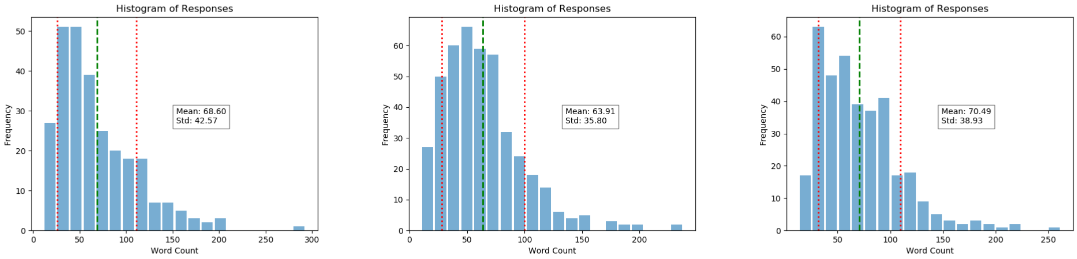
(d) Grade 3 (e) Grade 4 (f) Grade 5

Figure 5.2: Word counts per grade in responses D18

Figures 5.2 and 5.3 showcase histogram plots for the word counts for each grade in the two datasets. A histogram is a type of graph that shows how a set of continuous data is distributed [36]. It is made up of a set of bars that represent the frequency distribution of the data. In these figures, the x-axis represents the range of values for the word counts, while the y-axis represents the frequency at which each value (number of SAs) occurs within that word count. The mean and standard deviations are also marked in the plots with dotted lines colored green and red respectively. From these figures, it can be observed that the standard deviations are high in relation to the means. This indicates that the word frequency in the responses can overlap between the mean and standard deviations for multiple grades, making it difficult to use answer length as a part of the grading scheme.



(a) Grade 0 (b) Grade 1 (c) Grade 2



(d) Grade 3 (e) Grade 4 (f) Grade 5

Figure 5.3: Word counts per grade in responses D19

Chapter 6

Implementation and Evaluation

This chapter covers the proposed approaches of two automatic grading and feedback systems with Information Retrieval (IR) and Text Mining (TM) techniques. Furthermore, two suggested implementations of these approaches are present and tested on the Norwegian datasets described in Section 5.1. The focus will be on the implementation and experimenting with the possible performance that can be achieved on these approaches on the Norwegian datasets by testing out several preprocessing and feedback generation techniques. The main experiment is to evaluate the performance of the grading and feedback implementations on the datasets, and the goal will be to look at the possibilities for these approaches.

6.1 Tools and Environments

The two approaches were implemented in Python[1]. Python is the most suitable programming language for this work because of its big leap in the Natural Language Processing (NLP) field. In addition, Python provides a simple and easy-to-use syntax, making it easier for developers to write code. Visual Studio Code[2] was chosen as the code editor to be used in the development of these approaches. This was chosen because it is one of the most popular IDEs, and includes several handy development and environment features like a debugger, syntax highlighting and integrated Git[3]. Visual Studio Code also allows users to fully customize the editor with extensions that expand the functionality of the IDE. Users can choose between numerous Python-specific extensions, such as code formatters and linters. Additionally, GitLab[4] was utilized for source code management. This was chosen since it is NTNU's official source code management provider.

6.2 Dataset Preparation

During the exploration face of the datasets described in Section 5.1, a list of obstacles that occurred in the datasets was described. To prepare the datasets for the two approaches that were implemented, however, a series of tests were conducted on multiple preprocessing and cleaning techniques to figure out how to prepare the datasets for optimal performance on the algorithms. Thus, this section

---

[1] `https://www.python.org/`

[2] `https://visualstudio.microsoft.com`

[3] `https://git-scm.com/`

[4] `https://gitlab.stud.idi.ntnu.no/`

describes and covers the process of selecting the appropriate preprocessing techniques and cleaning methods.

6.2.1 Lexical Analysis

Firstly, to handle the obstacles found in the datasets, the proposed solutions to these obstacles that were explained in Section 5.1 were applied. From there, the first step in preparing the datasets was lexical analysis. The lexical analysis consists of first tokenizing all answers in the datasets, perform case folding and punctuation removal. The process included handling characters such as digits, hyphens and punctuation marks. This was performed by utilizing the NLP regex tokenizer[5] which utilizes regular expressions to recognize words, and at the same time remove punctuation marks and other unimportant characters. This is a more efficient way to tokenize in comparison to other tokenizers which simply splits text based on whitespace. Additionally, the choice of only keeping tokens that did not start with a number and were longer than three letters was made. This was because the two datasets consist of answers in the field of computer science and some responses contained programming snippets. Such snippets, when tokenized, resulted in a series of tokens that had few letters and contained no semantic meaning on their own. These tokens only contributed to an unnecessary amount of dimensions and were therefore chosen to be removed. The casing of the letters was also set to lowercase. This entailed that words like ‘Function’ and ‘function’ was treated as equal strings, and thereby equal tokens.

6.2.2 Stopword Removal

After lexical analysis, the removal of stopwords was performed. These words have little meaning to the context of the text and are therefore not necessary to have in the datasets. To remove these stopwords, the NLTK library was used. NLTK provides support for the removal of Norwegian stopwords by providing a corpus that includes a stopword list in Norwegian, and it functions by checking if each token matches the words in the list of stopwords. If the token matches a stopword, the word is removed. A visual inspection of the data revealed that not all stopwords were removed in this process. To further improve the stopword removal process, a list of Norwegian stopwords was manually constructed and applied as an algorithm to remove additional stopwords. Three open-source lists of Norwegian stopwords were found and utilized. The full list of stopwords utilized can be found in Appendix A.

6.2.3 Stemming

Stemming was performed after stopword removal. By employing stemming, words that contained the same root were grouped, and the dimensionality got reduced. The NLTK snowball stemmer[6] was utilized for Norwegian text, which is an iterative algorithm that takes an input word, converts it to lowercase and removes any diacritical marks. From there, the algorithm checks and removes common suffixes from the word. If the stem ends with a double consonant or the words are of a certain type, additional checks are performed. The stemming step resulted in a dimensionality reduction, and the majority of words (or tokens) were stemmed correctly. However, after a visual inspection of the two datasets, it was obvious that the stemming technique did not perform optimally in the Norwegian language.

---

[5] `https://www.nltk.org/api/nltk.tokenize.regexp.html`

[6] `https://www.nltk.org/api/nltk.stem.snowball.html`

6.2.4 Lemmatization

As with stemming, lemmatization was performed after the stopword removal step in an attempt to reduce the dimensionality. This was performed by loading a Norwegian language model that had been trained on Norwegian news and media text[7]. The model was able to utilize different functions on Norwegian text, such as lemmatization[8]. As a result, the words were grouped by their dictionary form, and thus, improved dimensionality. This corresponds well with relevant literature, as it is found that lemmatization has higher accuracy than stemming [44]. A combination of applying both stemming and lemmatization on the datasets was attempted to check if the performance increased. Performing both of these preprocessing techniques on the text has been shown to provide a slight increase in performance, where it is normal to perform lemmatization before stemming. However, in the case of these two datasets, lemmatization was found to perform the best singularly.

6.2.5 Synonym List

The usage of a synonym list was tested to group together words of the same meaning. This list was retrieved by an open-source GitHub repository containing a list of common Norwegian synonyms [40]. The repository retrieved synonyms from ‘Norske synonymer blå ordbok’ [18], and the file was structured by one headword with a list of synonyms beneath it. Stated differently, the synonyms were ‘grouped’ by the common headword. A function was implemented to check for synonyms in the two datasets and replace them with their corresponding headword in the synonyms file. This hypothesis would optimally group together words of the same meaning and further increase performance. The function loops through all terms in the two datasets, checks if the term is present in the synonym list, and replaces the term with its headword if the term is present in the list. Listing 6.1 presents pseudocode of this function:

```
def find_synonyms(text):
    # Load the norwegian synonyms from JSON
    synonyms = open(’norwegian-synonyms.json’, encoding=’utf-8’)
    textlist = text.copy()
    synonyms_data = json.load(synonyms)
    for i, word in enumerate(text):
        if word in synonyms_data:
            textlist[i] = synonyms_data[word][0]
    return textlist
```

Listing 6.1: Find synonyms function

The results of this implementation gave minor changes. The two datasets contain a combination of Norwegian language and English terminology, causing important terms to not get grouped together due to the English language. The implementation of an English synonym list was also attempted. However, this caused Norwegian words to be assigned as English words, changing their meaning and further decreasing performance.

---

[7] `https://spacy.io/models/nb`

[8] `https://spacy.io/api/lemmatizer`

6.2.6 Spell Correction

Spell correction was attempted to further group words of the same meaning. The data exploration phase revealed that several answers contained words of wrong spelling, causing the lemmatization and stemming steps to function inefficiently. Since the responses contain words in both English and Norwegian, an approach involving detecting what language the misspelling was in before suggesting the correct spelling was attempted implemented. This was performed by first defining a Norwegian and an English dictionary using the Python library 'Phunspell'[9]. 'Phunspell' is a Python library that provides a Pythonic API for communicating with the Hunspell spellchecker engine. Hunspell, however, is an open-source spellchecker engine that is widely used in several popular applications[10]. By offering a Pythonic API for communicating with the Hunspell engine, 'Phunspell' enables Python developers to quickly implement spellchecking capabilities into their Python projects. This makes 'Phunspell' a powerful and efficient spellchecking library for Python, with support for multiple languages including English and Norwegian.

The spell correction approach first checks if a word is in the Norwegian or English 'Phunspell' dictionary. If it is, the word is spelled correctly and is not changed. If the word is not in either of these dictionaries, the language detector 'Langdetect'[11] is applied. 'Langdetect' is a Python library for language detection based on statistical analysis of character n-gram frequencies. Almost 55 languages, including English and Norwegian are supported for language detection. The detector attempts to detect which language the misspelled word is in. If the misspelling is in Norwegian, the 'Phunspell' function 'suggest()' is used to return a suggested spelling of the misspelled word based on the Norwegian dictionary. If the misspelling is in English, the 'suggest()' function proposes a spelling based on the English dictionary. The word is not changed if no detection is present. This is to ensure that the algorithm suggests spellings in the correct language.

When evaluating the spellchecked tokens, it was discovered that they had little to no beneficial effect on the performance because the proposed spell checking algorithm contained some limitations. Norwegian words such as 'funksjonsscope' got changed to 'funksjonsperiode', which is a different word than intended. Furthermore, the implementation was not able to detect the languages of the misspelled words perfectly, most probably because of short inputs (one and one token separately) to the detect function. This made the approach not able to correct a misspelled Norwegian word like 'uvikle', because the 'Langdetect' library thought this word was in the language Lithuanian.

Feature Matrix

The input in the Reference Answer and k-means clustering approaches is a feature matrix, in other words, a Term-Document Matrix, which has each row representing TF-IDF weights in a SA and each column representing a feature (a term present in the response collection). The approaches can then calculate the distance between data points and centroids by using the Euclidian distance[12]. An example of how the Term-Document Matrix appears is presented in Listing 6.2.

```
                             Akkurat Bruke   Jquery
                                0      0      0.16
                                0     0.16    0.16
                                0     0.33    0.16
```

---

[9] https://pypi.org/project/phunspell/

[10] https://github.com/hunspell/hunspell

[11] https://pypi.org/project/langdetect/

[12] https://www.cuemath.com/euclidean-distance-formula/

```
                                  0        0        0
                                  0       0.41     0.14
                                 0.34      0       0.25
                                  0        0       0.09
                                  0        0        0
```

Listing 6.2: Example of Term-Document Matrix

The TfIdfVectorizer[13] from Scikit-learn is used to create the Term-Document Matrix. This vectorizer is adaptive and gives the user various options like ignoring terms that have a document frequency strictly lower or higher than a given threshold. In addition, the user can specify attributes like max features, if the term counts should be binary, and if the vectorizer should use n-grams. Therefore, when constructing the Term-Document Matrix, different approaches for the term representation were attempted, like n-grams of lengths 1-3. However, using n-grams lead to overly sparse matrices, which can lead to the matrixes consuming a large amount of memory and the computational efficiency may be affected. A solution to this is to ignore words that appear in less than 10% of the documents. This can simply be performed by setting the min_df parameter to 10% of the length of the dataframe in the 'TfIdfVectorizer'. In addition, the 'TfIdfVectorizer' automatically performs normalization on the feature weights by default, so there is no need to perform additional normalization on the matrices. This is done by applying L2 normalization on the matrices. L2 normalization is a technique used to normalize a vector by dividing it by its Euclidian norm [47]. The L2 norm of a vector X is defined as the square root of the sum of the squares of its components. Equation 6.1 illustrates how the L2 normalization is represented mathematically.

$$|\mathbf{x}| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{6.1}$$

6.2.7 Dimensionality Reduction

A dimensionality reduction was performed on a subset of tests on the two approaches in an attempt to further increase performance. TF-IDF weights were generally used, and consequently, a large set of words in the datasets would have low weights. Therefore, some words have lower relevance than others, and by removing such words, the performance might improve. This was performed by simply removing all words with TF-IDF weights under a given threshold.

6.3 Reference Answer Approach: Grading Results

The Reference Answer (RA) based approach was described in a high-level format in Section 4.2. For all tests and ideas utilized in this approach, the following Python libraries were used for the implementation: Pandas[14], Polyglot[15] and Scikit-learn[16]. The results of the first approach vary greatly with the choice of Reference Answers utilized and different preprocessing techniques. Consequently, to explore what quality can be achieved with such an approach, multiple choices of Reference Answers along with different preprocessing techniques had to be tested out. Thus, the results presented in this section are categorized as tests with varying choices of Reference Answers. All tests are separated by Reference Answers (RAs) and Student Answers (SAs), and the performance is computed by a Pearson correlation coefficient between the grades assigned by a teacher (Manualscore) on a

---

[13] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

[14] `https://pandas.pydata.org/docs/`

[15] `https://polyglot.readthedocs.io/en/latest/`

[16] `https://scikit-learn.org/stable/user_guide.html`

given SA and the average cosine similarity score that is calculated from the SA and RA(s). A further descriptive computation of the performance calculation is given for each test.

6.3.1 Test 1: All Highest-Graded Answers

Initially, an idea of an approach was to have the highest-graded answers in the datasets act as RAs. The rest of the datasets (answers graded 4 and under) would act as Student Answers. After all preprocessing steps and construction of the Term-Document Matrix were done, similarities could be calculated between the SAs and RAs. Since multiple RAs were used, a weighted average was performed on the similarities computed on one SA. This way, the average similarity of all the RAs was computed.

Naturally, the performance of such an approach can vary depending on the preprocessing steps that were performed. The performance, consequently, differed when lemmatization, stemming, spell correction and synonym lists all were performed individually and in combination with each other. A list of all preprocessing techniques that were performed and tested on different approaches is presented in Section 6.2.

A dimensionality reduction was performed in this test to further increase the performance of the algorithm. A threshold was set to remove words that had TF-IDF weights lower than 0.2. This way, the dimensionality was reduced substantially and the similarity measure became more accurate by removing irrelevant words.

A simple check to evaluate the performance of such an approach is to compare the average cosine similarity measure on one SA with the Manualscore a teacher has given on the answer. The Pearson correlation method was utilized for this purpose. After the algorithm was performed on a dataset and average cosine similarities were computed, correlations could get calculated between the cosine similarities and the Manualscore columns. Table 6.1 presents the correlation coefficients for all combinations of preprocessing steps that were tested for this approach on both datasets.

Table 6.1: Performance on all highly graded answers as Reference Answers

Preprocessing combination	**D18 performance**	**D19 performance**
Stemming	0.36309	0.42804
Lemmatization	0.36999	0.42133
Lemmatization + stemming	0.36979	0.42531
Lemmatization + synonym list	0.37311	0.41924
Spell correction + lemmatization	0.37391	0.42579
Spell correction + lemmatization + synonym list	0.37737	0.42098

As presented in the table above, all combinations of preprocessing techniques were very similar in performance. In addition, all correlations for this approach are regarded as rather low, varying from 0.36309 to 0.42804. The average performance across the two datasets was approximately 0.396 for all preprocessing combinations, with negligible variance. The reasons for this low correlation can be many, but the quality and the reliability of the two datasets are to be regarded as the main source. The datasets contain a combination of Norwegian language and English terminology. Furthermore, questions and answers that contain programming may cause the similarity and correlation scores to drop, simply because there are multiple ways of expressing oneself when two languages are combined. In addition to programming, it makes it hard for the preprocessing techniques to identify similar languages correctly. The two datasets' reliability and analysis are discussed in Section 7.1.

Another bias is the distribution of the datasets. Using all of the highest-graded answers as RAs causes only answers graded 4 and below to be evaluated. This may cause a skewed distribution of the data because there are no highly graded answers to be evaluated. Consequently, high similarity scores may not be present because all high-quality answers are used as RAs.

6.3.2 Test 2: Top 10 of the Highest-Graded Answers

Instead of employing all of the highest-graded answers as RAs, an idea was discovered to utilize only ten of the highly graded answers. This way, the distribution of grades in the SAs would increase because they would consist of a set of answers graded five. Additionally, employing ten of the highly graded answers would theoretically grant a variety of terms used in highly graded answers to be utilized as reference terms. If fewer highly graded answers were to be utilized as RAs, however, the variations of terms to be compared with would be smaller and the similarity calculations would hypothetically become more strict.

As with the first test, the performance varies depending on the preprocessing steps taken. Furthermore, a dimensionality reduction was also performed here, by removing all words with a TF-IDF weight lower than 0.2. Table 6.2 presents the results achieved on this test when utilizing various preprocessing tasks.

Table 6.2: Performance on ten highly graded answers as Reference Answers

Preprocessing combination	**D18 performance**	**D19 performance**
Stemming	0.36670	0.43502
Lemmatization	0.37107	0.43930
Lemmatization + stemming	0.36837	0.43387
Lemmatization + synonym list	0.37467	0.43875
Spell correction + lemmatization	0.37361	0.44569
Spell correction + lemmatization + synonym list	0.37453	0.44317

The performance on this test improved from the previous test in Section 6.3.1 with an average performance of 0.41, where the combination of spell correction and lemmatization performed the best. However, the correlation is still to be regarded as somewhat low, although a Pearson correlation of 0.41 can be regarded as moderately high.

For this approach, a bias might occur where some questions in the datasets have a varying amount of highly graded answers. Selecting ten of the highly graded answers may not be a good pick for some of the questions because the questions might not have ten, or a low amount of highly graded answers. Thus, the variations of performance on the questions might be high.

6.3.3 Test 3: Half of the Highest-Graded Answers

The number of answers graded five on each question varies, as explained in Section 5.2.1. Consequently, the number of highly graded answers to be used as Reference Answers should also vary from question to question depending on the number of highly graded answers existing on that question. This way, the number of Reference Answers used would vary dynamically per question. The idea was that the grade distribution in the Student Answers would be more evenly distributed if the same proportion of highly graded answers are left as Student Answers per question. The performance would also hypothetically improve this way.

To dynamically change the number of Reference Answers used in each question, a decision was made to utilize half of the highly graded answers as Reference Answers per question and leave the rest of the highly graded answers as SAs. Table 6.3 presents the results from this test.

Table 6.3: Performance on half of the highly graded answers as Reference Answers

Preprocessing combination	**D18 performance**	**D19 performance**
Stemming	0.38019	0.44346
Lemmatization	0.38230	0.44111
Stemming + lemmatization	0.38260	0.44411
Lemmatization + synonym list	0.38545	0.43755
Lemmatization + spell correction	0.38101	0.44843
Lemmatization + spell correction + synonym list	0.38296	0.44314

The performance on this test gave a slight increase from the two former tests. The 2018 dataset performance increased to an average of 0.38274 and the 2019 dataset to 0.44239. The total averages 0.41051, which is a slight increase from the former approach. The reason for this slight increase might be that the number of RAs differs dynamically. In addition, the number of highly graded answers to be used as SAs also varies depending on the question. Consequently, the grades on the SAs might become more evenly distributed.

6.3.4 Test 4: Most Common Terms in the Corpus

Instead of simply choosing Reference Answers from the two datasets, an interesting test would be to have a set of important terms act as RAs. Here, a set of the most common terms across the responses for each question is chosen to act as a reference. This is performed by extracting n terms with the overall highest TF-IDF weights across the corpus for each question. The performance here would differ on the number of words extracted, along with the preprocessing techniques performed on the data.

For this test, two vectors will be correlated with each other per SA. One vector contains the n common terms in the corpus, while the other is a vector of equal length containing a binary representation of which of the common terms appear in the SA. A choice was made to use a binary weight approach for this test instead of representing the words in the two vectors as TF-IDF weights. Hence, the SA vector would consist of 0 and 1's, where 1 indicates that the term is occurent in the SA. This approach simply wishes to check how many common words in the corpus are apparent in the SA and generate a similarity score, and not take into consideration how often they occur. Table 6.4 presents the performance on this test with varying preprocessing combinations. Furthermore, the number of most common terms used in the vector varies from 60, 80, and 100.

Table 6.4: Performance on most common words in the corpus used as reference

Preprocessing combination	**D18**			**D19**		
No. of terms	**60**	**80**	**100**	**60**	**80**	**100**
Stemming	0.47151	0.46946	0.46705	0.51413	0.50790	0.50632
Lemmatization	0.46962	0.47834	0.46789	0.50669	0.50241	0.50106
Stemming + lemmatization	0.46594	0.47219	0.46432	0.51675	0.50610	0.50389
Lemmatization + synonym list	0.47125	0.47589	0.46381	0.50220	0.49913	0.50117
Lemmatization + spell correction	0.46810	0.48131	0.46940	0.50889	0.50615	0.50195
Lemmatization + spell correction + synonym list	0.46783	0.47634	0.46444	0.50481	0.50188	0.50252

The performance on this test has overall increased from the three former tests. For the 2018 dataset, using 80 common words gave the highest average performance of 0.47 across all preprocessing steps. The same occurred for the 2019 dataset, with an average of 0.50 with 80 common words.

6.3.5 Honorable Mention

The four tests described in the sections above are the RA tests that gave the highest performance on this approach. Multiple other tests have also been performed but resulted in a lower performance than the ones presented above. One of these was to use one suggested solution answer as RA. The idea was to have one thorough, well-written, and descriptive suggested solution for each question. This way, all highly graded answers would be kept as SAs, providing a better grade distribution, and the suggested solution would be descriptive enough to give a wide variety of terms for comparison. However, the results gave an average performance of less than 0.4 on all combinations of preprocessing steps, causing the test to underperform in comparison to the other tests.

6.4 Reference Answer Approach: Feedback Generation Results

The goal of the feedback generation approach is to generate as high-value feedback as possible to improve learning for students. The Reference Answer approach described in Section 6.3 utilizes highly graded answers or the most common words in the corpus as RA's to compute similarity to SA's. Similarly, the words used in the RA's can be used to generate feedback for students, depending on what is missing in their responses. The RA's are always answers or words of high value, featuring important terminology and concepts that should be contained in a high-value SA. An idea is that the most common words used in the RA's will work as possible feedback words to students. Regardless of what text is used as RA's, the most common words in these responses can be extracted. Table 6.5 presents an extraction of a subset of common words from the first question in the 2018 dataset. 20 of the most common words in the RA's have been chosen to be extracted, along with common words from three SA's with varying grades. This table is an illustration of the terms that can be utilized as feedback. The terms have been retrieved by employing the most common words in the corpus to act as RAs, elaborated on in Section 6.3.4, and with lemmatization as preprocessing technique.

Table 6.5: Feedback terms and Student Answers From Q1 on the 2018 dataset

Candidate	**Terms**
Top 20 words from RA's	scope, deklarert, variable, funksjon, gjelde, blokk, global, definere, variabel, utenfor, block, bruke, tilgjengelig, console, nøkkelord, inne, endre, innenfor, lokal, deklarere
Student 1	block, blocke, console, definere, deklarert, error, funciton, function, funksjon, gjelde, inne, scope, scopefor, test, tilgjenlig, variabe, variable
Student 2	avhengig, blokk, deklarert, enten, funksjon, funskjon, gjelde, global, inne, javascrip, løkke, scope, settning, utenfor, variable
Student 3	blokk, brack, curly, deklarert, funksjon, funksjonsscope, gjelde, hel, igjen, initialisere, inne, iterere, løkk, løkke, nær, omvendt, scope, tilgjengelig, ulik, utenfor, variabel

Once common terms have been extracted from the RA's, they can be used to compare with SA's. An algorithm can simply check for the presence of common words in the SAs. If a word is not present, it might be a useful feedback term to the student. To improve the effectiveness of such an algorithm, the SAs should naturally be preprocessed and, perhaps, important words be extracted for comparison. Listing 6.3 presents the pseudocode of the feedback generation/word matching algorithm.

```
def get_feedback_terms(high_grade_terms, candidate_terms):
    feedback_terms = []
    for term in high_grade_terms:
        if term not in candidate_terms:
            feedback_terms.append(term)
    return feedback_terms
```

Listing 6.3: Feedback generation function

The algorithm takes a set of high-value terms from the RAs that can possibly be employed as feedback as a parameter. These words are in this case the most common terms derived from the set of RAs, but the function would work on any set of high-value terms sent as parameters. The second parameter is composed of candidate terms that are present in one SA. These words could either be high-value terms, terminology or simply the whole SA. As an illustration, two rows from Table 6.5 consisting of RA and SA terms can be utilized as parameters. The algorithm simply checks for the presence of RA words in the set of SA words. If the word is absent in the SA, it will be appended to a list and possibly used as feedback to the student. Naturally, SAs vary in the way they are written and the type of words they use. Therefore, sufficient preprocessing steps must be taken beforehand to limit the variety of words and synonyms as much as possible for the algorithm to work optimally.

When the algorithm is applied to a set of SAs, the result is a list of terms that are possible feedback terms. Table 6.6 illustrates the results when the algorithm is applied to the three before-mentioned Student Answers.

Table 6.6: Terms retrieved from feedback

Top 20 words from RA	**Candidate**	**Candidate terms**	**Retrieved feedback terms**
scope, deklarert, variable, funksjon, gjelde, blokk, global, definere, variabel, utenfor, block, bruke, tilgjengelig, console, nøkkelord, inne, endre, innenfor, lokal, deklarere	Student 1	block, blocke, console, definere, deklarert, error, funciton, function, funksjon, gjelde, inne, scope, scopefor, test, tilgjenlig, variabe, variable	blokk, global, utenfor, bruke, tilgjengelig, nøkkelord, endre, innenfor, lokal, deklarere
	Student 2	avhengig, blokk, deklarert, enten, funksjon, funskjon, gjelde, global, inne, javascrip, løkke, scope, settning, utenfor, variable	definere, variabel, block, bruke, tilgjengelig, console, nøkkelord, endre, innenfor, lokal, deklarere
	Student 3	blokk, brack, curly, deklarert, funksjon, funksjonsscope, gjelde, hel, igjen, initialisere, inne, iterere, løkk, løkke, nær, omvendt, scope, tilgjengelig, ulik, utenfor, variabel	variable, global, definere, block, bruke, console, nøkkelord, endre, innenfor, lokal, deklarere

The performance of this feedback approach greatly depends on the word usage of the students. Hence, appropriate preprocessing techniques to reduce the vocabulary of the students might increase the performance of the feedback algorithm. As seen in Table 6.6, a portion of feedback terms retrieved mean the same as the terms that the candidates actually used. The terms 'blokk' and 'block' mean the same but are in different languages. Therefore, the feedback the students receive might be biased, and an evaluation measure must be applied. Furthermore, not all terms represent an important concept of what should be in a high-quality answer. Thus, calculating the precision of the feedback algorithm is one way of evaluating the performance. This can be achieved by comparing the number of correct feedback terms generated to the total number of feedback terms retrieved by the algorithm. A choice was made to define a correct feedback term as a term that (1) represents an important concept that is relevant to the question and (2) has no synonymous meaning from any other term in the SA. By utilizing these criteria, a manual inspection can be performed to calculate the precision for the three candidates in Table 6.6. The precision of the three candidates is showcased in Table 6.7.

Table 6.7: Feedback precision

Candidate	**Precision**
Student 1	3/12
Student 2	4/13
Student 3	6/11

This method of evaluating the feedback is fairly straightforward and can be performed on a larger scale across the whole dataset. The manual inspection enables a method of clearly investigating which words actually fulfill the criteria for a feedback term. However, the goal of providing feedback to students was to improve learning, and it is hard to evaluate the learning outcome of this technique without gathering data from these students. To investigate how the feedback generation of this approach may increase learning, further evaluation should be performed. User testing of the implementation, conducting interviews, or a questionnaire to gather quantitative data may help in the evaluation task.

6.5 Clustering Approach

The clustering approach encompassed employing the k-means algorithm on the two datasets. The Python library Scikit-learn was utilized for this purpose[17]. This library is mainly developed for machine learning and provides various algorithms and tools for classification, regression, clustering and dimensionality reduction tasks. Scikit-learn offers a consistent and user-friendly interface for handling machine learning models, which is one of the main reasons for this selection. Furthermore, it integrates well with other popular Python libraries for scientific computing, such as NumPy[18], Pandas[19] and Matplotlib[20]. This enables the users to import, modify and display data quickly and simply using standard tools and processes.

Splitting the Dataset

Before applying k-means clustering, it could be beneficial to divide the Term-Document Matrix into a test and train set. This is to evaluate how well the k-means clustering algorithm performs on new unseen data, by having it trained on one portion of the datasets and tested on another. By splitting the data into a test and a train set, the performance can be evaluated based on the trained data, providing a better sense of how well the algorithm will generalize to new data.

Another reason to split the data is to prevent overfitting. This may occur when the algorithm is trained on the whole Term-Document Matrix, leading to weak performance on new data. By separating the data, the train set may be leveraged to determine the optimal value for k (number of clusters), and then use the test set to assess the algorithm's effectiveness. This can help avoid overfitting and improve the performance of the algorithm when new data is added.

The k-means algorithm learns from the train set by calculating the optimal centroid locations for the clusters that exist in a particular training set. These clusters are found under the training of the algorithm and can be utilized to predict cluster labels for new unseen data points (data points

---

[17] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

[18] `https://numpy.org/doc/stable/`

[19] `https://pandas.pydata.org/docs/`

[20] `https://matplotlib.org/stable/users/index`

in the test set) based on which centroid they are closest to. This implies that the algorithm does not recalculate the centroids when operating on the test dataset, as it is only assigned to the cluster with the closest centroid based on cluster distances from the training phase.

Fitting the Model

The model utilized for clustering is an unsupervised k-means machine learning model. When fitting (constructing matrixes) the model, it is necessary to keep the training and test sets separate. This means that only the training data should be utilized for any preprocessing operations, such as fitting a TF-IDF vectorizer. This is because the model should only learn from the training data, whereas the test data should only be employed for performance evaluation. If a subset of the test data is preprocessed, the model learns about the test data before it is ready to predict its labels, which may cause overfitting and poor generalization of new unseen data. The ‘fit_transform()’ from the aforementioned ‘TfIdfvectorizer’ is only applied to the training data to fit the vectorizer onto the training set, in order to transform it into a numerical representation. This representation is then utilized to train the k-means model. Then, the test data is transformed into the same numerical format as the training data by utilizing the ‘transform()’ method from the ‘TfIdfvectorizer’ without refitting the vectorizer. This ensures that the test data is represented in the same manner as the training data, without leaking any information about the test data to the model. To sum it up, data leakage and overfitting may occur if the vectorizer is fitted to the entire dataset, as the vectorizer would incorporate information from the test set, which can lead to biased results.

The ‘TfIdfVectorizer’ recalculates the TF-IDF weights of the words in the test set based on the document frequencies of each word in the train set. However, it does not re-fit the IDF (inverse document frequency) weights because the purpose of IDF weighting is to reflect the importance of a term in the context of the entire corpus or dataset used for training. In other words, the vectorizer only needs to learn the vocabulary of the training data and calculate IDF values for the train set. Then, these values are applied to the test set with its corresponding TFs to construct TF-IDF weights.

Finding Optimal Numbers of Clusters

Before applying the k-means clustering approach, a function that calculates the optimal number of clusters to separate the responses as much as possible was developed. This function is applied prior to the clustering approach itself. It initiates by receiving a dataframe that contains already preprocessed SAs. Next, the function iterates over a selected range of cluster values that are to be tested. For each cluster, the function splits the data into a test and train set, fits the vectorizer on the train data, and transforms both sets into numerical features. It then constructs a k-means model with the current number of clusters and fits the model onto the training matrix. Subsequently, the function predicts cluster labels for each row in the test matrix using the fitted k-means model and then calculates the average silhouette score for that particular cluster. The function ends by outputting each cluster and the corresponding silhouette score, along with the optimal number of clusters highlighted in the output. Listing 6.4 presents simplified pseudocode for this function.

```
def find_optimal_num_clusters(df):
    for n_clusters in range(2, 12):
        X_test = vectorizer.transform(test_df['lemmatized'])
        kmeans = KMeans(n_clusters=n_clusters, random_state=52)
        kmeans.fit(X_train)
        test_labels = kmeans.predict(X_test)
        silhouette_avg = silhouette_score(X_test, test_labels)
        silhouette_scores.append(silhouette_avg)
    best_n_clusters = silhouette_scores(max(silhouette_scores)) + 2
    print("Best number of clusters:", best_n_clusters)
```

Listing 6.4: Find optimal number of clusters function

Overview

An overview of the clustering approach is presented in Figure 6.1. The figure initiates with a dataset being passed into the NLP preprocessing pipeline. From there, a Term-Document Matrix for each question in the dataset is constructed. Then, the clustering algorithm trains on 70% of a matrix in order to find the optimal cluster centroids for the data. Based on this, the algorithm clusters new responses that originate from the last 30% of the dataset. The top N most common words in each cluster can be extracted, for then to be compared against common words in a RA to investigate which terms or concepts a group of responses has forgotten to mention. Based on this, the system can suggest common feedback for the given group of responses. Such a system is to be regarded as semi-automatic, where the suggested cluster feedback can provide an indication of what grade the responses in a given cluster are in. In a full-scale solution and after the feedback generation step, a teacher should be able to display groups of responses in a clear way. Then, step in and modify the common grading and feedback suggested by the system, as mentioned in Section 4.3.

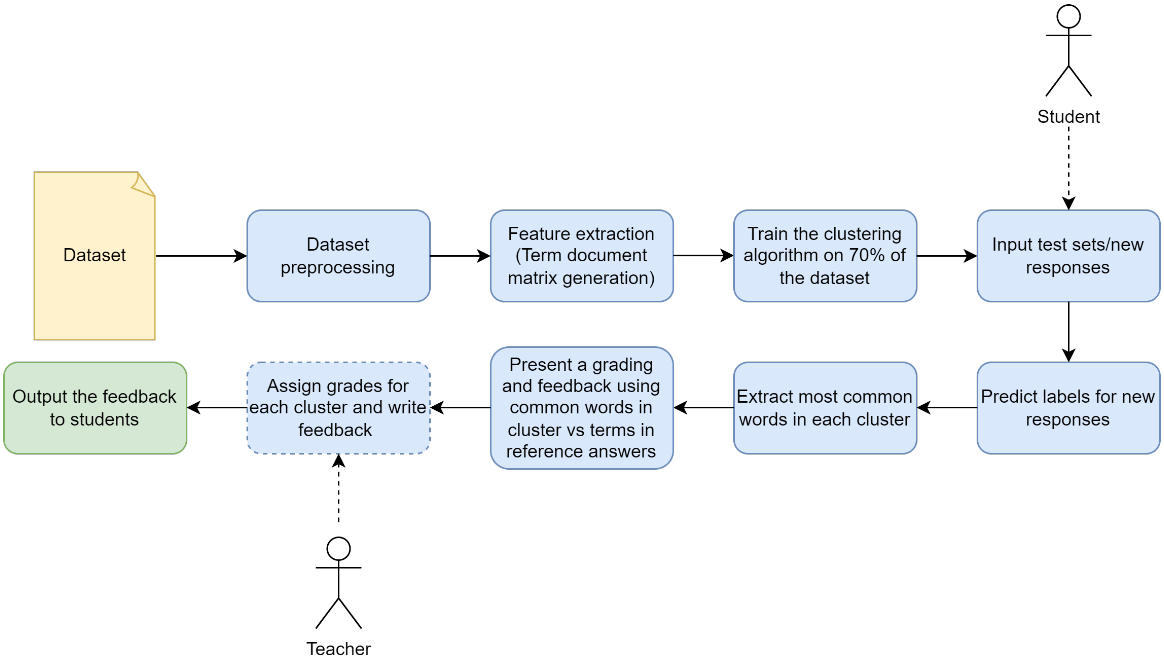

Student
Dataset
Dataset preprocessing
Feature extraction (Term document matrix generation)
Train the clustering algorithm on 70% of the dataset
Input test sets/new responses
Predict labels for new responses
Extract most common words in each cluster
Present a grading and feedback using common words in cluster vs terms in reference answers
Assign grades for each cluster and write feedback
Output the feedback to students
Teacher

Figure 6.1: Overview of the clustering approach

6.5.1 Grading Results

Before the k-means clustering was utilized, the dataset was split (30% - 70%) using the train_test_split function from Scikit-learn[21]. Then, the vectorizer was fitted on the training set. The vectorizer was designed to ignore words that appear in less than 10% of the responses and to incorporate unigrams, bigrams and trigrams. The test data was transformed into the same numerical representation as the training data, but without refitting the vectorizer, as explained in Section 6.5. After this, the k-means algorithm was run and the labels for the test data were predicted. As mentioned, this approach is to be regarded as semi-automatic. The common feedback in a cluster can be used as an indication of the grade, or manually chosen by a teacher. More specifically, by comparing the common cluster terms with potential Reference Answer terms, an indication of the common grade in a given cluster can be made. With this said, the main focus of this approach is on the feedback generation part.

One question was chosen for each dataset to evaluate the results of the clustering approach. More specifically, question one from D18 and question six from D18. Furthermore, one possible RA are listed for each question. These questions and RAs are listed below:

D18 Q1: Forklar kort hvilket scope som gjelder for variabler som er deklarert med nøkkelordet var og hvilket scope gjelder for variabler deklarert med nøkkelordet let?

D18 Q1 RA: *Var har funksjons-scope og er synlig innen hele funksjonen de er deklarert i. let har blokk-scope og er kun synlig innenfor blokken de er deklarert i. Begge er globale hvis de deklareres utenfor en blokk.*

D18 Q6: Forklar hva funksjonen groupBy gjør og gi et eksempel på bruk gitt cars-variabelen? Vi er ute etter overordnet funksjonalitet og bruk – og ikke detaljene i kallet til reduce.

D18 Q6 RA: *Funksjonen lar deg gruppere objekter basert på en navngitt property/key. Dette er implementert som en funksjon som tar key som input og returnerer en funksjon som tar en array av objekter som input. Førstnevnte brukes for å «sette opp» grupperingsfunksjonen. Sistenevnte brukes for å kjøre selve grupperingen.*

Figures 6.2 and 6.3 present the frequency of each student response's Manualscore in each cluster when ran with six clusters on the two aforementioned questions. Six clusters were chosen since there exist six different scores. The frequency of Student Answers is shown on the y-axis and the cluster ID's on the x-axis. The different colors in the bars represent the different Manualscores given by the sensors, and an explanation for the different colors can be seen in the labels on the right side of the plots.

---

[21] `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

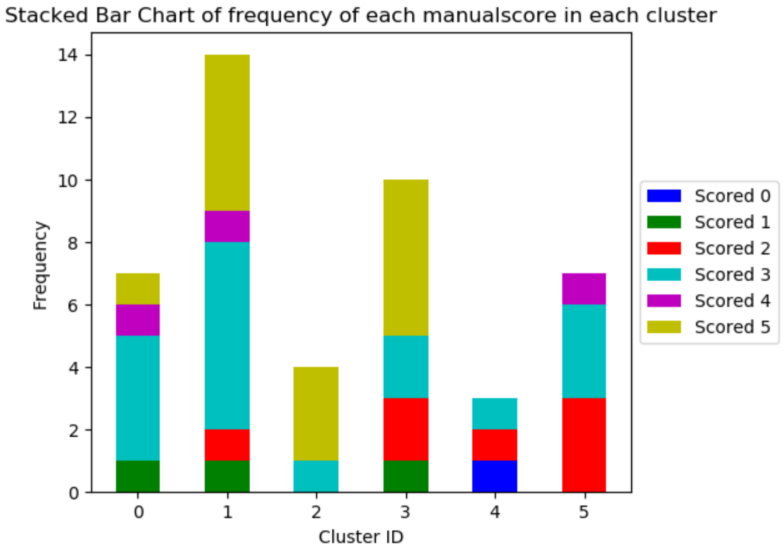


Figure 6.2: Manualscores in different clusters D18 Q1

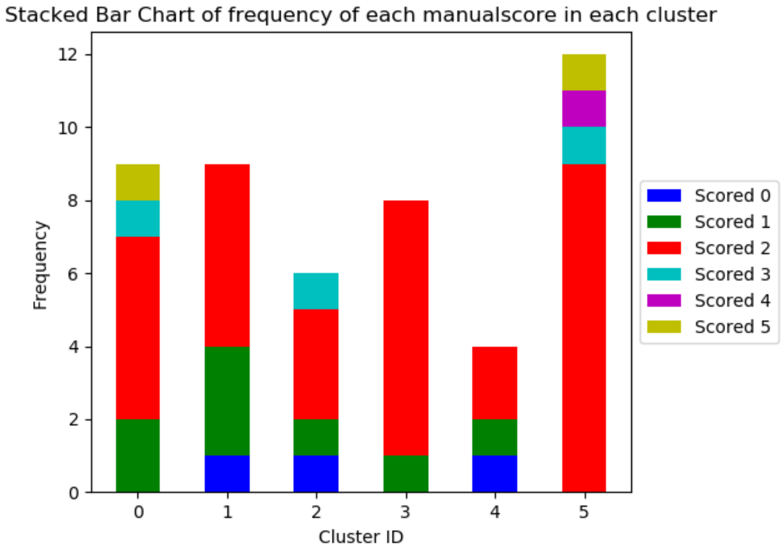


Figure 6.3: Manualscores in different clusters D18 Q6

The purity scores (Equation 2.2) were calculated to see how well the clusters match the Manualscores. Table 6.8 presents an overview of the purity scores for each cluster in the two plots. From these tables, it can be seen that the k-means algorithm does not perform very well in separating the different Manualscores. Only three clusters have a purity score of 0.75 or higher, while the overall purity for both questions is 0.5 and 0.62. Two clusters also have a purity score under 0.5. This indicates that the algorithm may perform poorly in separating different Manualscores into separate clusters.

Table 6.8: Purity scores for D18 Q1 and D18 Q6

Clusters	**D18 Q1**	**D18 Q6**
Overall Purity	**0.50**	**0.62**
0	0.57	0.55
1	0.43	0.55
2	0.75	0.5
3	0.5	0.88
4	0.33	0.5
5	0.43	0.75

It is also important to look at more than just the purity when evaluating the clustering. This is because the purity only looks at how well the clusters match the ground truth labels (Manualscores). It does not take into consideration the structure of the clusters, like how well separated or overlapped the clusters are. Purity is also sensitive to class imbalances, where one class is more represented than the others. This is the case for these datasets as they contain a substantially larger amount of highly graded answers than lower graded answers. Additionally, the purity scores increase if the clustering algorithm is run with a higher number of clusters, even if the number of clusters does not make sense for the context of the task. Because of this, the silhouette score (Equation 2.3) was also investigated during the evaluation phase of the approach. The silhouette score can tell more about the structure of the clustering, meaning how well separated or overlapped the clusters are. From Table 6.9, it is seen that the silhouette scores are above 0, indicating that the objects in the clusters are more similar to their own clusters than to other clusters. Additionally, these silhouette scores indicate that the clusters are somewhat distinct and meaningful, and the scores are decent considering the issues in the datasets.

Table 6.9: Overall silhouette scores for clustering D18 Q1 and D18 Q6.

Clustering	**Silhouette Score (6 clusters)**
D18 Q1	0.138
D18 Q6	0.097
Clustering	**Silhouette Score (optimal num clusters)**
D18 Q1	0.17
D18 Q6	0.13

In general, the overarching goal is not necessarily to cluster together responses with the same Manualscores in a grading and feedback system. One may assume that a grading system ideally wishes to cluster together similar responses before grades have been set. This way, a system can automatically find the optimal number of clusters utilizing the function described in Section 6.5. Then, the system and teacher can cooperate to provide general feedback to the clustered responses, regardless of how many clusters there are. When the 'find_optimal_num_clusters' algorithm runs on D18 Q1 and D18 Q6, the function suggests that 9 clusters are best on D18 Q1 and only 2 clusters for D19 Q6.

With these numbers of clusters, the silhouette score increases as seen in Table 6.9. This indicates that the data tends to be more separated and less overlapping when the number of clusters suggested by the 'find_optimal_num_clusters' function is applied, which is preferred in such a system.

The clusters can be visualized using a TSNE plot[22]. TSNE is a tool to visualize high-dimension data. In such a plot, the x and y values represent the two-dimensional embeddings of the data points. The TSNE algorithm reduces the high-dimensional data into two dimensions, which can be plotted on a 2D scatter plot. Each response is represented by a point in the scatter plot, and each point's location on the plot is defined by its own dimensional embedding obtained from the TSNE algorithm [20]. Points that are close to each other on the scatter plot are more similar to each other than points that are far apart. In the scatter plots, the points (responses) in each cluster are color-coded based on their assigned cluster labels. Figures 6.4 and 6.5 illustrate the TSNE plots for D18 Q1 and D19 Q6 when the optimal numbers of clusters are chosen.

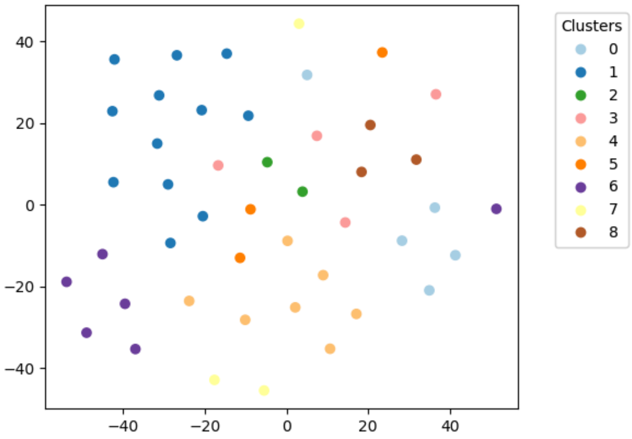


Figure 6.4: TSNE plot D18 Q1

---

[22] `https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html`

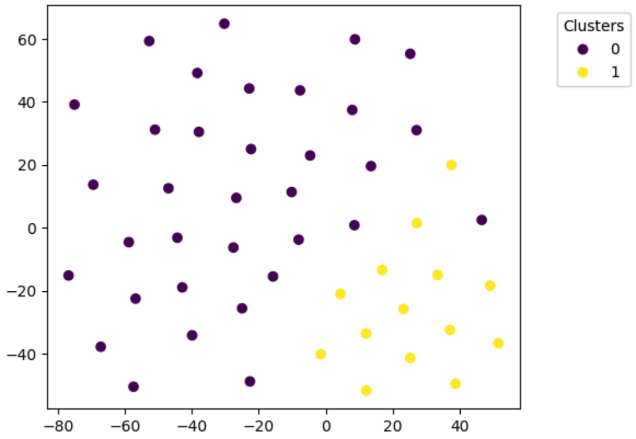


Figure 6.5: TSNE plot D19 Q6

These scatter plots reveal that, in general, most of the clusters are well-defined but a few outliers exist. For the D18 Q1 TSNE plot (Figure 6.4), it can be seen that clusters one, two, four and eight are well-defined without any outliers far away. This is highlighted in Figure 6.6. Cluster 3 is spread out around clusters two and eight, while clusters zero, five, six and seven are somewhat well-defined, but contain one outlier far away from the well-defined clusters. This indicates that the similarities in clusters one, two, four and eight are probably higher than in the remaining ones. In the D19 Q6 TSNE plot, the two clusters can easily be spotted. Only one outlier is present for cluster 0 which is far more to the right than the cluster itself. Silhouette scores for each cluster are summarized in Table 6.10.

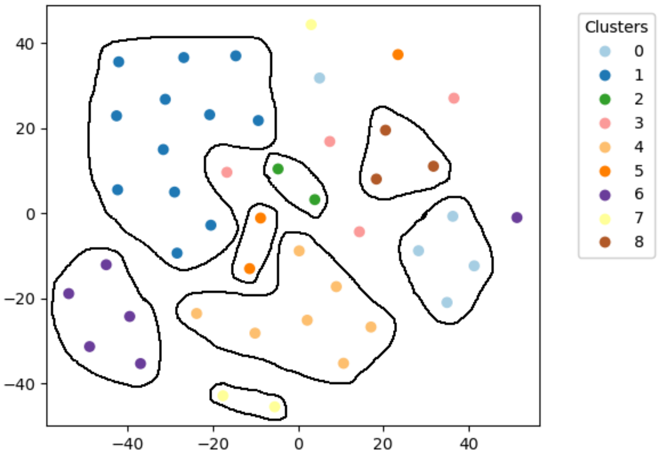


Figure 6.6: TSNE plot D18 Q1 clusters highlighted

Table 6.10: Silhouette scores for each cluster in D18 Q1 and D18 Q6

Clusters	**D18 Q1**	**D19 Q6**
0	0.157	0.023
1	0.157	0.388
2	0.279	-
3	0.01	-
4	0.076	-
5	0.095	-
6	0.303	-
7	0.374	-
8	0.206	-

6.5.2 Feedback Generation Results

One method to generate feedback to students clustered together in the same clusters was developed for the clustering approach. Providing automatic common feedback on responses that are clustered together might be a useful technique for providing efficient and consistent feedback to students. In this approach, top N frequent words are extracted from each cluster of student responses and compared against common terms in Reference Answers (RAs). By identifying the overlap between two sets of terms, the system can generate automated feedback that highlights areas where students may be struggling or making common mistakes. The system may offer targeted feedback to address this issue, for instance, if a set of students in a cluster use the same incorrect term or concept. This method can spare time for teachers and provide students with feedback more efficiently. However, it is essential that the RAs are correct and that the clustering algorithm has been correctly calibrated to prevent false positives or negatives. Furthermore, it is important to utilize human grading/evaluators to overlook the grading and feedback generated by the clustering algorithm to guarantee accuracy and fairness in the grading and feedback process.

Common terms are extracted from each cluster by first sorting the terms in the cluster centers in descending order so that the highest-scoring terms are listed first. Listing 6.5 showcases pseudocode for the top N terms extraction in N clusters. These terms are sorted using the Numpy 'argsort()' function[23], and the indices are saved in a variable called order_centroids. Then, the actual terms associated with those indices are retrieved using the 'get_feature_names_out' function from the vectorizer. This way, the top N terms for each cluster can easily be retrieved.

```
N=10
order_centroids = kmeans.cluster_centers_.argsort()[:, ::-1]
terms = vectorizer.get_feature_names_out()
for i in range(num_clusters):
    top_N_words = [terms[term] for term in order_centroids[i, :N]]
    print("Cluster {}: {}".format(i, top_N_words))
```

Listing 6.5: Top N term extraction function

The top 10 most frequent terms in each cluster for D18 Q1 and D18 Q6 are presented in Tables 6.11 and 6.12 respectively. From the D18 Q1 terms, it can be seen that some of the same terms appear in different clusters, such as 'global', 'scope', and 'variable'. This can indicate that these terms are

---

[23] https://numpy.org/doc/stable/reference/generated/numpy.argsort.html

important and may also exist in the question text. In addition, the clusters have unique terms that differentiate them. Examples of such terms are ‘console’ in cluster six and ‘løkke’ in clusters three and five. It can also be observed that clusters zero, three, four, five, seven and eight contain the same term ‘funksjon’, and that clusters zero, one, two, four, six, seven and eight contain the same terms ‘blokk’ or ‘block’. ‘Funksjon’ and ‘blokk’ are important terms in this question and an essential part of a high-scoring response. Only clusters zero, four, seven and eight contain both terms, indicating that the responses in these clusters are not far away distance-wise from the RA.

Table 6.11: 10 most frequent terms in each cluster D18 Q1

Clusters	**Features**
0	‘definere’, ‘scope’, ‘deklarert’, ‘variable deklarert’, ‘variable’, ‘global’, ‘blokk’, ‘funksjon’, ‘lokal’, ‘global scope’
1	‘deklarert’, ‘variable deklarert’, ‘variable’, ‘scope’, ‘block’, ‘innenfor’, ‘block scope’, ‘global’, ‘gjelde’, ‘global scope’
2	‘gjelde’, ‘global’, ‘scope’, ‘variable’, ‘global scope’, ‘hel’, ‘lokal’, ‘blokk’, ‘eksempel’, ‘block scope’
3	‘tilgjengelig’, ‘deklarere’, ‘variable’, ‘global’, ‘deklarert’, ‘funksjon’, ‘inne’, ‘scope’, ‘løkke’, ‘lokal’
4	‘block’, ‘funksjon’, ‘scope’, ‘utenfor’, ‘block scope’, ‘gjelde’, ‘innenfor’, ‘bruke’, ‘inne’, ‘global’
5	‘variabel’, ‘scope’, ‘eksempel’, ‘utenfor’, ‘løkke’, ‘funksjon’, ‘deklarert’, ‘hel’, ‘innenfor’, ‘tilgjengelig’
6	‘console’, ‘blokk’, ‘global’, ‘scope’, ‘inne’, ‘bruke’, ‘definere’, ‘løkke’, ‘innenfor’, ‘global scope’
7	‘bruke’, ‘variable’, ‘variabel’, ‘innenfor’, ‘funksjon’, ‘løkke’, ‘eksempel’, ‘global’, ‘scope’, ‘block scope’
8	‘blokk’, ‘scope’, ‘funksjon’, ‘deklarert’, ‘bruke’, ‘variable deklarert’, ‘gjelde’, ‘variable’, ‘utenfor’, ‘løkke’

For the 10 most frequent terms in each cluster for D18 Q6 (Table 6.12), it can be observed that only the terms ‘brand’ and ‘audi’ are common in both clusters. Cluster zero contains more general terms related to arrays, objects, functions and grouping of data. Examples are ‘array’, ‘objekt’, ‘liste’, ‘funksjon’ and ‘groupby’. Conversely, cluster one includes more general terms related to the example code in the exam question text such as ‘brand’ and ‘color’. This may indicate that the responses in cluster zero have succeeded in explaining the main part of the question to a greater degree, namely explaining the overall functionality and usage of the ‘groupby’ function.

Table 6.12: 10 most frequent terms in each cluster D18 Q6

Clusters	**Features**
0	‘array’, ‘objekt’, ‘liste’, ‘funksjon’, ‘groupby’, ‘cars’, ‘audi’, ‘verdi’, ‘brand’, ‘farge’
1	‘color’, ‘brand’, ‘color white’, ‘white’, ‘audi color’, ’brand audi color’, ‘brand audi’, ‘audi’, ‘white brand’, ‘color white brand’

When the clustering algorithm has finished and these terms have been extracted, common feedback can be presented to students. To evaluate the feedback provided to students based on common terms in the clusters, a test may be performed by employing a sample set of student replies. This can be achieved by comparing the common terms in the clusters to top N terms in the high-graded (scored 5) responses to generate feedback for each cluster. This way, we outsource the highest quality terms in each cluster based on highly graded answers. When the difference between the top N terms in the high-graded answers versus the common terms in the clusters are retrieved, the rest of the student responses can be compared against this difference. The resulting terms from this test can be divided into four sub-groups depending on the relevance of the term and used to evaluate the effectiveness of the algorithm:

- True Positive (TP): When the feedback mentions that the cluster mentions an important term, and the response also does so.
- False Positive (FP): When the feedback mentions that the cluster mentions an important term, but the response does not.
- False Negative (FN): When the feedback mentions that the cluster is missing an important term, but the response includes this term.
- True Negative (TN): When the feedback mentions that the cluster is missing an important term, and the response also does so.

Showing what similar responses have been written about as feedback to students can have several benefits. First, it can identify common mistakes by identifying frequent errors done by students and deliver targeted feedback to help them correct these errors. It can identify additional comments or find patterns that the instructor might have missed. Furthermore, it can encourage self-reflection and improvement in students. Students may be motivated to reflect on their work and find areas where they can improve by seeing what other students have done well and what mistakes they have made. This can also give students a sense of how well they understand the subject compared to other students. This technique would help evaluate the effectiveness of the feedback part of the system. The feedback also includes, at the end of the feedback, the terms that are missing in the specific student's response in comparison with the high-grade terms.

To sum it up, the proposed student feedback can be separated into two main parts, both having convenient use cases for the students. Firstly, high-grade terms from the same cluster that the student is placed in are presented. These are terms that similar responses include, along with terms that these responses are commonly missing. Secondly, terms retrieved from highly graded answers that are not present in the response are also presented, independent of the cluster formation. This way, the student would hypothetically receive three lists of terms that could be useful. An example is presented below of how such feedback could look like:

Similar answers like yours have written about important concepts like this: ***[Scope, funksjon, variable]****. Similar answers like yours are commonly missing this:* ***[block, innenfor]****. You might have missed this:* ***[deklarert, innenfor]****.*

Table 6.13 showcases responses from three students in question 1 in D18, all in different clusters and received different Manualscores. In addition, proposed feedback is given and each term from the feedback is categorized into the evaluative sub-groups described above.

Table 6.13: Feedback test

Cluster	**Scored**	**Response**	**Feedback**	**Test result**

0	1	[‘scope’, ‘deklarert’, ‘innover’, ‘kode’, ‘nøstinge’, ‘loope’, ‘mistenke’, ‘bruke’, ‘utenfor’, ‘klasse’, ‘funksjon’, ‘definere’, ‘usikke’, ‘forskjell’, ‘bruke’, ‘hverandre’, ‘problem’]	Liknende svar som ditt har skrevet om viktige aspekter som dette: *[‘scope’, ‘funksjon’, ‘deklarert’, ‘blokk’, ‘variable’, ‘variable deklarert’, ‘definere’]*,
			men mangler å skrive om: *[‘block’, ‘utenfor’, ‘innenfor’]*
			Dette bør du kanskje skrive mer om: *[‘blokk’, ‘variable’, ‘variable deklarert’, ‘block’, ‘innenfor’]*
				block = true negative
				utenfor = false negative
				innenfor = true negative
				scope = true positive
				funksjon = true positive
				deklarert = true positive
				blokk = false positive
				variable = false positive
				variable deklarert = false positive
				definere = true positive

1	5	[‘variable’, ‘deklarert’, ‘function’, ‘scope’, ‘variable’, ‘deklarert’, ‘block’, ‘scope’, ‘bruke’, ‘utenfor’, ‘funksjon’, ‘global’, ‘variabel’, ‘funksjon’, ‘block’, ‘scope’]	Liknende svar som ditt har skrevet om viktige aspekter som dette: *[‘scope’, ‘deklarert’, ‘variable’, ‘variable deklarert’, ‘block’, ‘innenfor’]*,
			men mangler å skrive om: *[‘funksjon’, ‘blokk’, ‘utenfor’, ‘definere’]*
			Dette bør du kanskje skrive mer om: *[‘blokk’, ‘variable deklarert’, ‘definere’, ‘innenfor’]*
				funksjon = false negative
				blokk = false negative
				utenfor = false negative
				definere = true negative
				scope = true positive
				deklarert = true positive
				variable = true positive
				variable deklarert = true positive
				block = true positive
				innenfor = false positive

5	2	[‘global’, ‘scope’, ‘lokal’, ‘block’, ‘variabel’, ‘lag’, ‘endre’, ‘hel’, ‘endre’, ‘datatype’]	Liknende svar som ditt har skrevet om viktige aspekter som dette: *[‘scope’, ‘funksjon’, ‘deklarert’, ‘utenfor’, ‘innenfor’]*,
			men mangler å skrive om: *[‘blokk’, ‘variable’, ‘variable deklarert’, ‘block’, ‘definere’]*
			Dette bør du kanskje skrive mer om: *[‘funksjon’, ‘deklarert’, ‘blokk’, ‘variable’, ‘variable deklarert’, ‘utenfor’, ‘definere’, ‘innenfor’]*
				blokk = false negative
				variable = true negative
				variable deklarert = true negative
				block = false negative
				definere = true negative
				scope = true positive
				funksjon = false positive
				deklarert = false positive
				utenfor = false positive
				innenfor = false positive

The ‘Test results’ column in Table 6.13 can be inspected to evaluate the feedback generation. It can be seen from the first response that the test results consist of six true and four false test results. This response was given a score of one by the grader. The response addressed several important terms such as ‘scope’, ‘funksjon’, ‘deklarert’, and ‘definere’ as the similar responses in this cluster also do. However, the response missed mentioning terms like ‘block’ and ‘innenfor’ which are important terms for a high-graded response. These were also mentioned by the feedback of what similar responses like this response was missing. Additionally, the feedback mentions that terms like ‘blokk’, ’variable’, and ‘variable deklarert’ exist in similar answers, but are not mentioned in the response. The term ‘blokk’ can be ignored, as it exists in the response, but in another language. Overall, the response is rather similar to the cluster it belongs to, and the student can take inspiration from the terms ‘variable’ and ‘variable deklarert’ to improve their response.

For the second response, which got a score of five, the ‘Test result’-column depicts five negative and five positive terms. The feedback mentions that the cluster is missing terms like ‘funksjon’ and ‘blokk’, but the response includes these terms in another language, indicating some language problems in the system. Furthermore, only one false positive and one true negative is present in this response, implying that the response includes most of the terms that the cluster includes. This indicates that this student may not take so much inspiration from the common terms in the cluster feedback. However, considering that the student got a grade of five on this response, it suggests that less inspiration is to be taken from the feedback.

The third response got a score of two, and the test results include four false positive terms. These terms are ‘funksjon’, ‘deklarert’, ‘utenfor’ and ‘innenfor’. These terms are all important for a high-graded response and were left out. Therefore, the student can utilize these terms to improve his response. The test results also include three true negative terms: ‘variable’, ‘variable deklarert’ and ‘definere’. These are terms that similar answers also missed and may be beneficial feedback terms for the student.

Lastly, it is important to note that in an ideal clustering grading and feedback system, a teacher/evaluator should check the feedback retrieved by the system, and perhaps write their own common feedback to the clusters if they disagree with the system or the system functions poorly.

6.6 Code Quality

The implementation of the two approaches presented in this chapter is built with the idea of scalability and reusability. Hence, if other datasets of the same structure were to be utilized on the two implementations, the code would function without requiring significant modifications or rewriting. This encompasses code for the two approaches, data cleaning, and data preprocessing. The idea is that datasets containing exam answers of unstructured text generated by the online Inspera platform would work with minimal modifications. Thus, other datasets containing text of various lengths and themes could also be used with the same code to test these approaches. Furthermore, parts of the code could be used singularly as modules for other tasks, such as only cleaning Inspera data or performing a few preprocessing techniques. Specific algorithms for single actions (such as finding synonyms and removing stopwords) are built in a way that they can be performed on any type of unstructured text that is sent as a parameter, independent of the textual context. Additionally, code for adding Reference Answers manually has also been implemented since both approaches rely on Reference Answers. Lastly, note that all code implemented is available upon request.

Chapter 7

Discussion

In order to further understand the results presented in Section 6, a discussion has to be made. This chapter first discusses the aspects of the datasets used in this thesis and their reliability, as it plays a considerable role in the results. Secondly, the chapter discusses the results presented by the Reference Answer and clustering approaches. Lastly, a few limitations regarding the work of this thesis are discussed.

7.1 Dataset Reliability

In order to evaluate the quality of automatic grading and feedback systems, it is important that the data that is fed into the system is appropriate and of high quality. Naturally, the data that gets fed into a system is a factor that determines the quality achieved by such systems. Thus, the reliability of the two datasets utilized in this thesis must be discussed.

One of the main differences that is present on the two datasets in contrast to datasets found in earlier work, is that they are mostly written in Norwegian. Furthermore, a combination of Norwegian language and English terminology were used in most of the answers. This, consequently, is an occurring problem that has not been identified by earlier work. A choice was made that only Norwegian Bokmål answers were to be utilized in this thesis, although the datasets contained Norwegian Nynorsk and English answers. By only utilizing Norwegian Bokmål answers, issues might occur when the system is applied in other languages. Furthermore, the combination of Norwegian answers and English terminology may cause several preprocessing techniques to underperform compared to a single-language dataset. Synonym lists and spell checking algorithms can struggle to understand multi-language answers and therefore cause translations that are incorrect and thus, decrease performance.

In the dataset analysis step, several features were investigated. It was found that the grades in the datasets, ranging from 0-5, were skewed in their distribution. Most answers were graded 3-5, while a subset was graded 0 and 1. Such a skewed distribution may cause the performance to differ, as it may provide a lack of graded answers to successfully train a clustering model, or too few answers to be used as a reference for comparison. The datasets also initially included code snippets by students. This was chosen to be removed and not be a part of the answers, as it complicates the evaluation of Student Answers significantly. This decision can be regarded as rather drastic as it might manipulate a set of Student Answers causing highly graded answers to be under-evaluated, or oppositely. Furthermore, the length of the answers was also investigated. It was found that the answer length increased slightly by grade on both datasets. Automatic grading and feedback systems may perform differently depending on the length of the answers. Thus, big differences in answer length may provide a bias in the grading and feedback system depending on the data that is being fed into the system.

Several preprocessing steps along with choices to remove parts of the datasets were performed. These steps are described in Sections 5.1 and 6.2. A choice was made to take a closer look at the lemmatization step and combine it with both synonym lists and spell correction. However, the stemming technique was not combined with any other technique other than stopword removal. This was because the stemming technique underperformed in comparison to lemmatization, and the fact that lemmatization is regarded as an overall better performing technique with higher accuracy than stemming in earlier work [45]. Other preprocessing techniques, combinations and critical choices regarding the dataset could therefore provide a better performance.

The aim of this thesis was to explore how an automatic grading and feedback system could be implemented with the use of Text Mining (TM) and Information Retrieval (IR) techniques, and what quality could potentially be achieved. With this in mind, the focus has been on Norwegian short answers from two exams in computer science. A list of obstacles in the datasets was presented in Section 5.1 which included Inspera formatted text, the presence of programming questions and special characters. With datasets containing such a variety of answers, it is hard to evaluate what quality that can be achieved in the two approaches implemented in this thesis. However, by focusing solely on natural text and trying to reduce or remove the amount of programming code, spelling mistakes and synonym words, an attempt to increase the quality of the two approaches on these datasets has been performed. It is hard to conclude whether the quality would increase if the programming questions were kept in the datsets, evaluated separately, or if fewer data-cleaning steps were taken.

7.2 Reference Answer Approach

The Reference Answer based approach encompassed calculating a similarity score on a Student Answer (SA) and a Reference Answer (RA) to predict a grade. In addition, a feedback generation technique based on the content of RAs was proposed. The quality of this approach is highly dependent on what was used as RAs and the preprocessing techniques applied. Thus, several variations of RAs were tested along with combinations of preprocessing techniques. The results varied more when different choices of RAs were chosen and less when variations of preprocessing techniques were utilized.

On all four tests performed on changing the type of RAs, the choice of preprocessing combination had little impact on the overall performance. The difference in performance when applying stemming, lemmatization, or both can be regarded as negligible. This may be because stemming and lemmatization are two very similar preprocessing techniques with the same goal. Stemming reduces a word to its canonical form, while lemmatization changes the word to its dictionary form. However, lemmatization gave a minor average increase in performance on all tests. Other combinations of preprocessing techniques varied slightly in performance, and these variances can also be regarded as negligible because the changes are too small to produce a significant impact. Moreover, these small differences are based on the two datasets used in this thesis, and the performance may slightly differ when using other datasets. Hence, concluding with one optimal preprocessing combination for all tests cannot be done because there is no combination that significantly outperforms the others. However, an optimal combination of preprocessing techniques might exist, but that requires further testing.

The first test in Section 6.3.1 utilized all highly graded answers as RAs. The idea was that this would provide a high variation of words to be used as references to the remaining SAs. However, the distribution of grades in the SAs got skewed by using all answers graded five as SAs. Consequently, the average performance came down to 0.4. By utilizing all the highly graded answers as RAs, the similarity score calculated would naturally perform poorly because there were no highly graded SAs left to calculate similarity against. Thus, splitting up the highly graded answers was proposed.

Section 6.3.2 presents the second test. Here, only ten of the highly graded answers were used as RAs in an attempt to average out the grade distribution in the datasets while remaining a high variation of words in the RAs. This slightly increased the performance from the first test, especially on the 2019 dataset. The main reason for this is that the SAs consisted of a larger amount of highly graded answers to be compared to RAs. This, consequently, caused the similarity score to vary to a larger degree based on the quality of the SA. Thus, increasing the correlation of the Manualscore and similarity measure. It is also worth mentioning, however, that the 2018 dataset had a very negligible difference in performance on the two tests.

The third test presented in Section 6.3.3 proposed a dynamic change in the number of highly graded answers to be utilized as RAs based on the number of highly graded answers present on the current question in the datasets. More specifically, half of the answers graded five in the datasets were to be used as RAs. This way, the number of highly graded answers remaining as RAs would always be the same proportion in each question. The performance of this test improved on both datasets in comparison to the former tests. The average performance on the 2019 dataset came out to 0.443. The reason for this increase in performance might be that on average, all questions got the same amount of highly graded answers, causing the grade distribution to change dynamically per question and therefore provide a better fit in highly graded answers and variations in word usage. The 2018 dataset performance increased slightly on all preprocessing combinations in comparison to the two former tests.

Instead of having existing SAs act as RAs, the fourth test elaborated on in Section 6.3.4 utilized the most common words across the corpus per question as RAs. This way, all highly graded answers were kept as SAs, and a set of the most commonly used words was employed as a reference against the SAs. The performance of this test significantly improved compared to the former, with the highest performance found at 0.5167. The performance varied slightly when using 60, 80, or 100 of the most common words in the corpus. However, all variations of this test performed better than the other tests. One reason for this is that all highly graded answers were kept as SAs. Furthermore, the most common words used as reference exist in the SAs. Thus, the performance would naturally increase when the same words are used to compare.

Overall, the fourth test in Section 6.3.4 outperformed the others, with the highest performance found at 0.5167 on the 2019 dataset, utilizing stemming and lemmatization, and 60 common words in the corpus as reference. This Pearson correlation is to be regarded as moderately strong, indicating a strong linear relationship between the variables (the cosine similarity computation and Manualscore). Such a performance is not unusual in earlier work described in Section 3. However, based on related work, one can conclude that better performance has been achieved in the evaluation of short-text answers. It is worth mentioning, regardless, that this performance was achieved on Norwegian text-based answers with a large amount of English terminology. Furthermore, there is little work to be found on such datasets, and therefore hard to compare these findings with work utilizing different techniques and other datasets.

The feedback generation approach proposed encompassed providing relevant keywords to students based on the missing themes, or concepts in their answers. The feedback was generated by a simple word-matching algorithm that extracted keywords from RAs and compared them to SAs. From there, keywords representing important concepts that the student did not have in his answer were retrieved and delivered as feedback terms.

The performance of the feedback approach is highly dependent on the word-matching algorithm. This is because the algorithm is responsible for extracting the correct feedback terms that are delivered to the students. Consequently, there exist multiple ways to improve the feedback algorithm. One such improvement could be to implement a synonym list or a pos-tag checker into the algorithm to check if the student is not using any synonymous term, or explanations that cover the same concept. That would not only aid in providing more precise feedback but also remove unwanted terms from the feedback generation. Furthermore, the word-matching algorithm utilizes terms from Reference Answers in this approach. One issue with this is that it is not certain that the full corpora of terms, or all concepts are covered in these Reference Answers for all questions. Thus, implementing some type of quality assurance, to ensure that all concepts are present in the word-matching algorithm as Reference Answers could be done.

The goal of the feedback approach was to generate as high-quality keywords as possible for feedback to the students to enhance learning. Thus, evaluating the feedback approach by looking at the retrieved feedback terms was necessary. The evaluation technique proposed in Section 6.4 was based on calculating the precision of the retrieved feedback terms. This was done by counting the number of correctly retrieved feedback terms divided by the total number of feedback terms retrieved. It is a rather simple approach that requires a manual inspection of the feedback generation algorithm. Furthermore, it is an evaluation technique that is based on certain criteria for what defines a correct feedback term. These criteria can change based on the evaluator, and might not be the best solution.

Another way of evaluating the feedback generation is to perform a quantitative study, as mentioned in Section 6.4. In this approach, participants could hypothetically get asked if the feedback terms retrieved from the algorithm based on a question and response were meaningful, helpful, or represented an important concept in some manner. This approach would generate opinions from a larger amount of people, which could make it easier to determine the overall performance of the feedback generation algorithm presented.

7.3 Clustering Approach

This approach, including splitting the dataset and finding the optimal numbers of clusters may function in real situations in schools or universities, but it depends on several factors to function optimally. An important factor is the centroid initialization. In this approach, the k-means++ initialization technique was utilized. This method aims to distribute the initial centroids evenly across the data space, which helps avoid the algorithm to converge to a suboptimal solution. The random_state parameter was also used to ensure that the random part of the k-means++ initialization, including randomly selecting the first centroids from the data points, was the same across multiple runs of the algorithm with the same data and parameters. This makes the results reproducible and easier to compare the algorithm with varying parameters.

Another significant factor is the number of K clusters. If the number of clusters is too small, the algorithm may not capture the underlying patterns in the data. Conversely, if the number of clusters is too high it may result in overfitting or small and meaningless clusters. For this approach, a function to find the optimal number of clusters was developed and is described in Section 6.5. The function utilizes the silhouette scores to find the best separation of the data and the optimal number of clusters. Based on Section 6.5.1, it can be seen that the overall silhouette scores increased by about 0.03 for both questions when the optimal number of clusters was chosen. This indicates that running such a function that finds the optimal number of clusters can help, but for these datasets and questions, the silhouette scores are not very high. The silhouette scores of 0.17 and 0.13 can be considered as a moderate clustering quality, suggesting that the clusters may not be very well separated, but not unseparated either.

In this case, the dataset contains some noise and overlapping patterns. Therefore, it can be said that a score of 0.17 and 0.13 is acceptable considering the structure of the datasets. It is also seen in Section 6.5.1 that the algorithm did not split the Manualscores very well. This is not necessarily a big problem, as the most important factor in this approach is how well the algorithm separates different responses. Therefore, the silhouette scores are the most important to look at.

Additionally, the chosen distance metric for the algorithm plays a considerable role. The metric used in this approach is the Euclidian distance. Other possible distance metrics are squared Euclidian distance, Manhattan distance and chi-square distance. It is important that the choice of distance metric corresponds with the characteristics of the dataset and the problem at hand. Euclidian distance is one of the most popular distance metrics used in k-means clustering and measures the straight-line distance between two points in a multi-dimensional space. Euclidian distance works well when the data is continuous, and the attributes are of a similar scale which is the case in the datasets used in this approach. Thus, the Euclidian distance was chosen as the distance metric for this approach.

L2 normalization was also applied when vectorizing the documents. This technique scales each vector to have a unit norm, meaning that the sum of the squared values of the vector equals one. This helps in addressing the issue of different scales along different dimensions and can assure that all dimensions equally contribute to the similarity or distance calculations in the k-means clustering algorithm.

The data distribution also plays a considerable role in the clustering results. The outcome of the clustering can be influenced by the underlying distribution of the data and the existence of noise or outliers. In the context of this approach, the data distribution is a bit skewed to the right, which may affect the centroid's final placement and the formation of the clusters.

The feedback generation presented for the clustering approach featured two main sections for feedback. Firstly, the top N high-grade terms within each cluster the specific SA belonged to were given. Secondly, terms occurring in highly graded answers that the response is missing independent of the cluster were presented. The argument for giving two forms of feedback was that the student could use these two parts in different ways to improve their answer. The top N high-grade terms in the clusters were given to identify common errors done by students. These terms could also identify additional patterns that a teacher might have missed, and students could encourage self-reflection by seeing what other students had done well and what mistakes they had made. As for the high-graded answer terms, they could be utilized as insights into concepts that the individual student might have missed, inspiration to improve learning, or simply an explanation of the grade that the specific student was given. Other alternative ways to generate the feedback could be to only present the top N cluster terms or only present the highly graded terms that the response is missing. However, this would affect the quality of the feedback in a negative way.

The feedback generation approach initializes with finding the top N words associated with each cluster. This function sorts the terms in the cluster centers based on their distance from the center of the cluster and returns the indices of the top N terms in each cluster. Once these indices are obtained, top N terms can be retrieved. This algorithm offers a simple and effective way to retrieve the top N terms in each cluster. It provides useful insights into the content of the responses within each cluster, which is an essential part of the generated feedback.

The feedback first suggests what important concepts the cluster mentions and what important concepts the cluster is missing. Alternative ways to provide feedback in a clustering system could be feedback that is more detailed and explanatory. Such type of feedback could include an explanation of the different terms that the cluster and the specific response are missing. For example, for a response that is grouped under a cluster that is missing important terms like 'Scope' and 'block', possible feedback could be:

Similar answers yours are commonly missing this: ***scope, block****. A variable that is declared with the keyword let has block-scope which means that the variable is only visible inside the block that it is declared in. Your response would offer a more comprehensive understanding of which scopes that applies for variables declared with var and let in JavaScript if you mention this.*

This would lead to somewhat longer feedback and could lead to students becoming disengaged and finding it difficult to extract the most important information from the feedback. Because of this, the developed feedback focuses on clear, concise and well-organized feedback that highlights the most important areas for improvement while avoiding irrelevant details. Students could also research the terms mentioned in the feedback themselves. This could engage the students more actively with the learning material, allowing them to develop their own understanding of the relevant terms.

7.4 Limitations

The work conducted in this thesis is based on two datasets featuring Student Answers related to computer science. Furthermore, it is in the Norwegian language. These criteria, consequently, generate a set of limitations related to the work of this thesis, and a discussion related to these limitations is in order.

First of all, Norwegian NLP is a few steps behind the English NLP field. This is mainly because there are more resources accessible for NLP development in the English language, such as large amounts of text data and well-established libraries. Norwegian, however, has a significantly smaller pool of text data, and less research has been conducted on the Norwegian language. Consequently, this leads to the Norwegian NLP field missing NLP resources, like an open synonym API for Norwegian text. The proposed solution for the work of this thesis encompassed utilizing an open-source GitHub repository containing a list of Norwegian synonyms that were retrieved from a Norwegian book called ‘Norske synonymer blå ordbok’ [18]. This is not an ideal resource for collecting synonyms, especially for grading computer science exams. A more appropriate approach could be to retrieve synonyms from multiple sources to build a common synonym list. Furthermore, it could be beneficial to develop a synonym list specified for computer science terms in this case. However, such a synonym list would be difficult to realize currently due to the lack of open synonym API’s for Norwegian text.

Another limitation was the time frame for this thesis. Multiple improvements for the systems were dropped due to this, such as a proper user test of the feedback generation. It was initially planned to conduct a questionnaire to evaluate the systems. This could provide valuable insights into how the feedback would be interpreted by the user, including whether it was helpful for the student or not.

Lastly, the fact that the approaches are only based on two datasets/exams is a limitation. For the clustering approach that groups similar student responses together, the accuracy can be affected by the amount of training data available. If ten exams had been utilized instead of two, the size of the training set would have been tremendously larger, potentially resulting in a much more accurate clustering algorithm. Similarly, for the Reference Answer approach, a bigger dataset including more exams could help the system to identify common patterns and themes with the use of a wider range of possible Reference Answers. This would help the system understand what a correct answer should include. Having a larger dataset would also help make the results of the Reference Answer approach more generalizable.

Chapter 8

Conclusion

The aim of this thesis was to explore techniques for implementations of automatic evaluation and feedback systems using Information Retrieval and Text Mining. Two Norwegian datasets featuring short-text answers have been utilized to evaluate these techniques and figure out what quality could potentially be achieved. Furthermore, two approaches for such systems have been proposed. This chapter covers a conclusion to the work. In addition, the contributions related to automatic grading and feedback this thesis presents will be summarized. Lastly, a discussion regarding future work in automatic grading and feedback generation will be presented.

8.1 Conclusion

Much research on automatic grading and feedback generation has been done, and it can be dated all the way back to the work performed by Page in the 1960s [31]. The majority of earlier work has focused on English answers, and little to no work is present on the Norwegian language. Thus, an investigation into automatic grading and feedback techniques in the Norwegian language can be seen as a new contribution within this field. For this thesis, three research questions were formulated in Section 1.2, and have been addressed throughout this thesis. This section will revisit these questions and attempt to formulate conclusions to them.

The first research question addressed the state of the art within automatic grading and feedback generation of short-text answers. The related work presented in Chapter 3 categorizes existing Automatic Short-Answer Grading (ASAG) systems into era's depending on the time of publication and implementation. These era's are Concept Mapping, Information Extraction, Corpus-Based Methods, Clustering and Machine Learning, each featuring different techniques for automatic grading of short-text answers. However, the common denominator for the systems of these era's is that they are based on the English language. The most relevant approaches for the goals of this thesis are also presented. These approaches feature systems for both automatic grading and feedback generation. One approach proposed by Rodrigues, Fatima et al. [38] is based on calculating the similarity between Reference Answers (RAs) and Student Answers (SAs). Furthermore, a feedback module is presented based on finding similar words in the RA and SAs. Another approach by Suzen et al. [41] is based on a clustering approach that clusters together similar answers into three clusters labeled Excellent, Mixed and Weak. Here, all answers in the same cluster received the same feedback.

The second research question encompassed how Information Retrieval and Text Mining techniques could be utilized to implement an automatic grading and feedback system. Chapters 2 and 3 cover a series of IR and TM techniques. Furthermore, they are applied for grading short-answer questions in a series of earlier works. Examples of such appliances are Natural Language Processing (NLP), Latent Semantic Analysis (LSA) and Term Frequency-Inverse Document Frequency (TF-IDF) which is used to evaluate a document to a query.

The third research question addressed how an automatic grading and feedback system could perform on real-world datasets. To answer this, two approaches for automatic grading and feedback systems were proposed and implemented. Furthermore, they were tested on two Norwegian exams featuring short-answer questions. The first approach was based on a similarity calculation between Reference Answers and Student Answers. It was found that the performance of this approach varied with the preprocessing techniques applied on the two exams and the choice of Reference Answers utilized. For this approach, the feedback technique encompassed giving keywords to students about concepts or themes that the students might have missed in their responses. An evaluation metric for the feedback generation algorithm was also proposed by calculating the precision of the given feedback terms.

The second approach for an automatic grading and feedback system was based on a clustering method. Here, the k-means clustering technique was applied to cluster together similar student responses. To achieve optimal grading performance, techniques such as splitting the dataset and finding the optimal number of clusters were applied. The proposed approach for grading responses in this approach encompassed comparing the most common terms in a cluster to some type of Reference Answer to get an indication of what grade the group of responses was in. The feedback generation method encompassed giving feedback terms in two forms:

1. Firstly, terms that similar student responses had in common and were part of the high-grade terms, and high-grade terms that the cluster was missing were given. In other words, high-value terms that occurred in the same cluster as the student response and high-value terms that the cluster did not mention.
2. Secondly, terms that the specific student response was missing in relation to high-grade terms were given.

The argument for providing two types of feedback terms was that they were beneficial for the students in different ways. Terms the specific student was missing could be beneficial to explain the grade a given student had been given. Terms from similar responses could be beneficial for inspiration and finding out what concepts the specific student had missed. Lastly, an evaluation method for the feedback approach was proposed by identifying the relevance of each feedback term retrieved. The terms could be divided into four sub-groups based on their relevance: true positive, false positive, false negative, and true negative.

8.2 Contributions

This thesis contributes to the field of automatic grading and feedback generation in the Norwegian language. Furthermore, two approaches for implementing such a system have been proposed and tested on two Norwegian exams in computer science.

First of all, a dataset analysis was performed on the two Norwegian exams. This analysis can be looked at as inspiration for analyzing, cleaning and processing other Norwegian text-based datasets or exams. The proposed use cases within this analysis and the steps taken in this study can provide insights into further text analysis. In addition, the methods for data preprocessing and cleaning utilized in this study can be performed and generalized to other types of unstructured text. Cleaning of multi-lingual datasets can also appear troublesome, and the techniques utilized in this analysis may prove useful. In particular, the implementation of a Norwegian synonym list and a multilingual spell correction algorithm are important contributions in this field that can be used as a starting point for further development. Furthermore, Text Mining and Information Retrieval techniques have been described and applied as a means to extract and evaluate text. Inspiration can be taken from these appliances.

A review of related work within this field is presented in Chapter 3. This review can prove beneficial for others seeking information or conducting further research. In addition to the background information presented in Chapter 2, these chapters can be utilized as an introduction to the fields of Information Retrieval and Text Mining, as well as some of the most relevant literature in these fields.

The main contributions of this thesis are the two proposed approaches for automatic grading and feedback generation. Information Retrieval and Text Mining techniques have been applied on two Norwegian text-based exams to investigate the possibility of a full-scale grading and feedback system. The proposed approaches, techniques and performance results can aid in the construction of such a system. Although the performance achieved on the two datasets in this study may not be sufficient, the results and techniques utilized are still highly relevant for further research and investigation. The main goal of this thesis was to explore techniques for automatic grading and feedback generation to investigate whether a full-scale system could be implemented with such techniques. Based on related work and the methods applied in this thesis, the possibility for such a system utilizing similar methods is highly relevant. Thus, the approaches and techniques presented in this study may be useful and can be utilized in a full-scale automatic grading and feedback system.

8.3 Future Work

This study has shown varying results with two different approaches in grading and providing feedback to Student Answers. However, there exists a lack of research on handling a mix of Norwegian and English languages in student responses for such systems. In order to further develop and improve automatic grading and feedback systems in different languages, especially in Norwegian, more research is necessary.

Further Testing on other Datasets and Subjects
The two datasets utilized in this work are from a computer science course that contained a mix of both Norwegian and English terminology, including code snippets. More testing on other Norwegian datasets could be done to provide a better understanding of how well the two proposed approaches in this thesis perform. Since the two approaches do not singularly focus on computer science datasets, additional testing could be performed on other topics to see if the exclusion of code snippets affects the performance positively. Furthermore, it would be interesting to see how the two approaches presented perform on datasets containing longer answers, such as answers of essay length.

Conduct a User Test along with Designing a Graphical User Interface (GUI) or Full-Scale Solution

In this research, the methods used for evaluating the generated feedback are performed manually. The goal of giving feedback to students is mainly to improve learning, and this is hard to evaluate without further evaluation of the two approaches. One way to do this is to conduct user tests. Such a user test could consist of conducting interviews, questionnaires, or developing a graphical user interface (GUI) of the two approaches that users could use to test the systems. Furthermore, it would be interesting to see how the two approaches proposed would work in a full-scale solution with a GUI. Such a user-friendly interface could make it easier for teachers and students to interact with and test the system. Additional capabilities that a GUI can offer include visualizations of a student's performance, progress over time, and options for customizing grading criteria and feedback. This can be of significant help for teachers to better understand their students' learning patterns. Moreover, a GUI could make it easier for non-technical users to use the system. This can improve the system's usability, making it more widely applicable in different areas of education.

Incorporate more Advanced Natural Language Processing (NLP) Techniques

The effectiveness and accuracy of an automatic short-answer grading system are highly dependent on the NLP techniques used. In this research, lexical analysis, stopword removal, stemming, and lemmatization were implemented, along with grouping together terms of the same meaning using a synonym list and spell correction. However, the implementation of the two latter techniques had almost no beneficial effect on the performance. Therefore, further research in these areas could be beneficial along with the incorporation of other advanced NLP techniques. Part-of-speech (POS) tagging is an important technique in NLP that can be leveraged to improve such systems. With POS tagging, each word in a sentence is given a unique part-of-speech such as a noun, verb, adjective, or adverb. By utilizing these techniques, the system can gain a deeper understanding of the meaning and structure of Student Answers, enabling it to identify errors that might otherwise go unnoticed.

Experiment with other Algorithms

Further testing of other algorithms is an important aspect of developing a better-performing grading and feedback system. There exist many different types of clustering algorithms that can be applied in such systems. Some of them are hierarchical clustering, spectral clustering, and density-based clustering. Each of these algorithms has the potential to increase the precision and efficiency of short-answer grading systems by grouping together more similar responses. Artificial intelligence methods, such as deep learning are also important concepts that can be utilized instead of similarity calculation. Such models can be trained on large datasets of human-graded short answers to build models that can recognize and interpret complex patterns in text and, furthermore, learn how to grade answers themselves. It would be interesting to see how the two datasets utilized in this study perform with these aforementioned approaches.

Support Multilingualism

As the datasets contained responses from different languages and a mix of both English and Norwegian in the same responses, handling this multilingualism is important for the performance of the system. In this research, English and Norwegian Nynorsk responses were excluded, as this thesis only focuses on Norwegian Bokmål answers. One possible approach for handling multiple languages in future systems can be to utilize machine translation algorithms to translate the student responses into one common language, e.g. English, before grading the response. However, this method can be challenging due to different languages having varied grammar, syntax and vocabulary. Pre-trained word embeddings can also be used to recognize the meaning of a word in different languages. This is done by providing a way to represent words in a high-dimensional vector space based on their semantic and syntactic properties. This method is also dependent on languages having common roots or similar grammatical structures to perform well.

Explore the Possibilities of Utilizing the GPT Models on Automatic Grading

The latest GPT models (3.5 and 4) are one of the most advanced language models in the world, as stated in Section 2.7.4. These models have had a significant impact on the field of NLP already. However, not much research has yet been conducted on how these models perform on grading and giving feedback to short-text answers. To utilize a GPT model for grading short-text answers, it would be a good idea to fine-tune the model on a specific dataset of short-text answers and their corresponding scores given by a teacher. Fine-tuning the model involves adjusting its parameters to optimize the grading performance. This is a big advantage, as it makes the model understand the patterns and features of the given datasets better. After training, the model can be leveraged to grade and offer feedback on student replies by taking the replies as input, followed by outputting a predicted grade and feedback to the user.

Bibliography

[1] Enrique Alfonseca and Diana Perez. “Automatic Assessment of Open Ended Questions with a Bleu-Inspired Algorithm and Shallow NLP”. In: Jan. 2004, pp. 25–35. ISBN: 978-3-540-23498-2. DOI: `10.1007/978-3-540-30228-5_3`.

[2] Fawad Ali. “GPT-1 to GPT-4: Each of OpenAI’s GPT Models Explained and Compared”. In: *MUO* (Apr. 2023). URL: `https://www.makeuseof.com/gpt-models-explained-and-compared`.

[3] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press / Addison-Wesley, 1999. ISBN: 0-201-39829-X. URL: `http://www.ischool.berkeley.edu/~hearst/irbook/glossary.html`.

[4] Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. “Powergrading: a Clustering Approach to Amplify Human Effort for Short Answer Grading”. In: *Transactions of the Association for Computational Linguistics* 1 (2013), pp. 391–402. DOI: `10.1162/tacl_a_00236`. URL: `https://aclanthology.org/Q13-1032`.

[5] Ashutosh Bhardwaj. “Silhouette Coefficient - Towards Data Science”. In: *Medium* (Dec. 2021). URL: `https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c`.

[6] Sridevi Bonthu, S. Rama Sree, and M. H. M. Krishna Prasad. “Automated Short Answer Grading Using Deep Learning: A Survey”. In: *Machine Learning and Knowledge Extraction.* Ed. by Andreas Holzinger et al. Cham: Springer International Publishing, 2021, pp. 61–78. ISBN: 978-3-030-84060-0.

[7] Jason Brownlee. “A Gentle Introduction to the Bag-of-Words Model”. In: *Machine Learning Mastery* (Aug. 2019). URL: `https://machinelearningmastery.com/gentle-introduction-bag-words-model`.

[8] Steven Burrows, Iryna Gurevych, and Benno Stein. “The Eras and Trends of Automatic Short Answer Grading”. In: *Int. J. Artif. Intell. Educ.* 25.1 (Mar. 2015), pp. 60–117. ISSN: 1560-4306. DOI: `10.1007/s40593-014-0026-8`.

[9] Jill Burstein, Susanne Wolff, and Chi Lu. “Using Lexical Semantic Techniques to Classify Free-Responses”. In: *Breadth and Depth of Semantic Lexicons.* 1996. ISBN: 978-94-017-0952-1. URL: `https://doi.org/10.1007/978-94-017-0952-1_11`.

[10] Sundaresh Chandran. “Gentle Introduction To Text Representation - Part - 1 | by Sundaresh Chandran | Towards Data Science”. In: *Medium* (Dec. 2021). URL: `https://towardsdatascience.com/introduction-to-text-representations-for-language-processing-part-1-dc6e8068b8a4`.

[11] Contributors to Wikimedia projects. *Computer-aided assessment - Wikipedia.* [Online; accessed 8. Dec. 2022]. June 2022. URL: `https://en.wikipedia.org/w/index.php?title=Computer-aided_assessment&oldid=1093745245`.

[12] Blaise Cronin. *Information Science and Technology.* asis&t, 2004.

[13] Laurie Cutrone, Maiga Chang, and Kinshuk. "Auto-Assessor: Computerized Assessment System for Marking Student's Short-Answers Automatically". In: *2011 IEEE International Conference on Technology for Education.* 2011, pp. 81–88. DOI: `10.1109/T4E.2011.21`.

[14] *Difference between Information Retrieval and Information Extraction - GeeksforGeeks.* [Online; accessed 23. Nov. 2022]. Jan. 2022. URL: `https://www.geeksforgeeks.org/difference-between-information-retrieval-and-information-extraction`.

[15] Lucas Galhardi and Jacques Brancher. "Machine Learning Approach for Automatic Short Answer Grading: A Systematic Review". In: Nov. 2018, pp. 380–391. ISBN: 978-3-030-03927-1. DOI: `10.1007/978-3-030-03928-8_31`.

[16] Rohith Gandhi. "Support Vector Machine — Introduction to Machine Learning Algorithms". In: *Medium* (Nov. 2022). ISSN: 9344-4447. URL: `https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47`.

[17] Suvarna Gawali. "Skewness and Kurtosis: Quick Guide (Updated 2023)". In: *Analytics Vidhya* (Feb. 2023). URL: `https://www.analyticsvidhya.com/blog/2021/05/shape-of-data-skewness-and-kurtosis`.

[18] Dag Gundersen. *Norske synonymer blå ordbok.* Kunnskapsforlaget, 2013.

[19] Christian Gütl. "e-Examiner: Towards a Fully-Automatic Knowledge Assessment Tool applicable in Adaptive E-Learning Systems". English. In: *International Conference on Interactive Mobile and Computer Aided Learning.* International Conference on Interactive Mobile and Computer Aided Learning ; Conference date: 18-04-2007 Through 20-04-2007. ., 2007, pp. xxx–xxx.

[20] Harshit. *t-SNE Algorithm in Machine Learning.* [Online; accessed 17. Mar. 2023]. Mar. 2023. URL: `https://www.enjoyalgorithms.com/blog/tsne-algorithm-in-ml`.

[21] Andrea Horbach, Alexis Palmer, and Magdalena Wolska. "Finding a Tradeoff between Accuracy and Rater's Workload in Grading Clustered Short Answers". In: *ACL Anthology* (May 2014), pp. 588–595. URL: `https://aclanthology.org/L14-1680`.

[22] Wen-Juan Hou et al. "Automatic Assessment of Students' Free-Text Answers with Support Vector Machines". In: vol. 6096. June 2010, pp. 235–243. ISBN: 978-3-642-13021-2. DOI: `10.1007/978-3-642-13022-9_24`.

[23] Arun Kanejiya Dharmendra. Kumar and Surendra Prasad. "Automatic Evaluation of Students' Answers using Syntactically Enhanced LSA". In: (May 2003).

[24] Ibrahim Abu El-Khair. "Term Weighting". In: *Encyclopedia of Database Systems.* Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 3037–3040. ISBN: 978-0-387-39940-9. DOI: `10.1007/978-0-387-39940-9_943`. URL: `https://doi.org/10.1007/978-0-387-39940-9_943`.

[25] Claudia Leacock and Martin Chodorow. "C-rater: Automated Scoring of Short-Answer Questions". In: *Computers and the Humanities* 37.4 (Nov. 2003), pp. 389–405. ISSN: 1572-8412. DOI: `10.1023/A:1025779619903`.

[26] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: `https://aclanthology.org/W04-1013`.

[27] Ph. D. Michel Kana. "Representing text in natural language processing - Towards Data Science". In: *Medium* (Dec. 2021). URL: `https://towardsdatascience.com/representing-text-in-natural-language-processing-1eead30e57d8`.

[28] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. "Corpus-based and Knowledge-based Measures of Text Semantic Similarity." In: vol. 1. Jan. 2006.

[29] Tom Mitchell et al. "Towards robust computerised marking of free-text responses". In: (Jan. 2002).

[30] Michael Mohler and Rada Mihalcea. "Text-to-Text Semantic Similarity for Automatic Short Answer Grading". In: *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*. Athens, Greece: Association for Computational Linguistics, Mar. 2009, pp. 567–575. URL: `https://aclanthology.org/E09-1065`.

[31] Ellis B. Page. "The Imminence of... Grading Essays by Computer". In: (1966), pp. 238–243. URL: `https://www.jstor.org/stable/pdf/20371545.pdf`.

[32] *pandas.DataFrame — pandas 1.5.3 documentation.* [Online; accessed 2. Feb. 2023]. Jan. 2023. URL: `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html`.

[33] Kishore Papineni et al. "Bleu: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: `10.3115/1073083.1073135`. URL: `https://aclanthology.org/P02-1040`.

[34] Diana Perez et al. "Automatic Assessment of Students' Free-Text Answers Underpinned by the Combination of a BLEU-Inspired Algorithm and Latent Semantic Analysis." In: Jan. 2005, pp. 358–363.

[35] ProjectPro. "Demystifying Transformers Architecture in Machine Learning". In: *ProjectPro* (Apr. 2023). URL: `https://www.projectpro.io/article/transformers-architecture/840`.

[36] Real Python. "Python Histogram Plotting: NumPy, Matplotlib, pandas & Seaborn". In: *Real Python* (Feb. 2023). URL: `https://realpython.com/python-histograms`.

[37] Dr.P.Selvi Rajendran and A. Bnerjee. "Automatic Short -Answer Grading System (ASAGS)". In: (Nov. 2010). URL: `https://arxiv.org/ftp/arxiv/papers/1011/1011.1742.pdf`.

[38] Fátima Rodrigues and Lília Araújo. "Automatic Assessment of Short Free Text Answers". In: *4th International Conference on Computer Supported Education* 2 (Apr. 2012).

[39] Dezső Sima et al. "Intelligent Short Text Assessment in eMax". In: *Towards Intelligent Engineering and Information Technology*. Ed. by Imre J. Rudas, János Fodor, and Janusz Kacprzyk. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 435–445. ISBN: 978-3-642-03737-5. DOI: `10.1007/978-3-642-03737-5_31`. URL: `https://doi.org/10.1007/978-3-642-03737-5_31`.

[40] Cathrine Stadsnes. "Evaluating Semantic Vectors for Norwegian". MA thesis. University of Oslo, 2018.

[41] Neslihan Süzen et al. “Automatic short answer grading and feedback using text mining methods”. In: *Procedia Computer Science* 169 (2020). Postproceedings of the 10th Annual International Conference on Biologically Inspired Cognitive Architectures, BICA 2019 (Tenth Annual Meeting of the BICA Society), held August 15-19, 2019 in Seattle, Washington, USA, pp. 726–743. ISSN: 1877-0509. DOI: `https://doi.org/10.1016/j.procs.2020.02.171`. URL: `https://www.sciencedirect.com/science/article/pii/S1877050920302945`.

[42] Skattum. Julian Trøan. Jesper. *Master_Preparatory_Project*. 2022.

[43] Shaun Turney. “Pearson Correlation Coefficient (r) | Guide & Examples”. In: *Scribbr* (Dec. 2022). URL: `https://www.scribbr.com/statistics/pearson-correlation-coefficient`.

[44] Kartik Wali. “Explained: Stemming vs lemmatization in NLP”. In: *Analytics India Magazine* (May 2022). URL: `https://analyticsindiamag.com/explained-stemming-vs-lemmatization-in-nlp`.

[45] Kartik Wali. “Explained: Stemming vs lemmatization in NLP”. In: *Analytics India Magazine* (May 2022). URL: `https://analyticsindiamag.com/explained-stemming-vs-lemmatization-in-nlp`.

[46] Hao-Chuan Wang, Chun-Yen Chang, and Tsai-Yen Li. “Assessing creative problem-solving with automated text grading”. In: *Computers & Education* 51.4 (2008), pp. 1450–1466. ISSN: 0360-1315. DOI: `https://doi.org/10.1016/j.compedu.2008.01.006`. URL: `https://www.sciencedirect.com/science/article/pii/S0360131508000341`.

[47] Eric W. Weisstein. “L2-Norm”. In: *Wolfram Research, Inc.* (Feb. 2002). URL: `https://mathworld.wolfram.com/L2-Norm.html`.

[48] *Welcome to polyglot’s documentation! — polyglot 16.07.04 documentation.* [Online; accessed 3. Feb. 2023]. Jan. 2021. URL: `https://polyglot.readthedocs.io/en/latest`.

[49] *What is Information Retrieval? - GeeksforGeeks.* [Online; accessed 3. Nov. 2022]. July 2022. URL: `https://www.geeksforgeeks.org/what-is-information-retrieval`.

[50] *What is Text Mining? | IBM.* [Online; accessed 14. Feb. 2023]. Feb. 2023. URL: `https://www.ibm.com/topics/text-mining`.

[51] *WordNet.* [Online; accessed 24. Nov. 2022]. Nov. 2022. URL: `https://wordnet.princeton.edu`.

[52] Soner Yıldırım. “Evaluation Metrics for Clustering Models - Towards Data Science”. In: *Medium* (Apr. 2022). URL: `https://towardsdatascience.com/evaluation-metrics-for-clustering-models-5dde821dd6cd`.

Appendix A

List of stopwords

NLTK corpus Norwegian stopwords:

- alle
- at
- av
- bare
- begge
- ble
- blei
- bli
- blir
- blitt
- både
- båe
- da
- de
- deg
- dei
- deim
- deira
- deires
- dem
- den
- denne
- der
- dere
- deres
- det
- dette
- di
- din
- disse
- ditt
- du
- dykk
- dykkar
- då
- eg
- ein
- eit
- eitt
- eller
- elles
- en
- enn
- er
- et
- ett
- etter
- for
- fordi
- fra
- før
- ha
- hadde
- han
- hans
- har
- hennar
- henne
- hennes
- her
- hjå
- ho
- hoe
- honom
- hoss
- hossen
- hun
- hva
- hvem
- hver
- hvilke
- hvilken
- hvis
- hvor
- hvordan
- hvorfor
- i
- ikke
- ikkje
- ingen
- ingi
- inkje
- inn
- inni
- ja
- jeg
- kan
- kom
- korleis
- korso
- kun
- kunne
- kva
- kvar
- kvarhelst
- kven
- kvi
- kvifor
- man
- mange
- me
- med
- medan
- meg
- meget
- mellom
- men
- mi
- min
- mine
- mitt
- mot
- mykje
- ned
- no
- noe
- noen
- noka
- noko
- nokon
- nokor
- nokre
- nå
- når
- og
- også
- om
- opp
- oss
- over
- på
- samme
- seg
- selv
- si
- sia
- sidan
- siden

- sin
- sine
- sitt
- sjøl
- skal
- skulle
- slik
- so
- som
- somme
- somt
- så
- sånn
- til
- um
- upp
- ut
- uten
- var
- vart
- varte
- ved
- vere
- verte
- vi
- vil
- ville
- vore
- vors
- vort
- være
- vært
- vår
- å

Open-source list one (https://gist.github.com/kmelve/8869818):

- alle
- andre
- at
- av
- bare
- begge
- ble
- blei
- bli
- blir
- blitt
- bort
- bra
- bruke
- både
- båe
- da
- de
- deg
- dei
- deim
- deira
- deires
- dem
- den
- denne
- der
- dere
- deres
- det
- dette
- di
- din
- disse
- dit
- ditt
- du
- dykk
- dykkar
- då
- eg
- ein
- eit
- eitt
- eller
- elles
- en
- ene
- eneste
- enhver
- enn
- er
- et
- ett
- etter
- for
- fordi
- forsøke
- fra
- fram
- før
- først
- få
- gjorde
- gjøre
- god
- gå
- ha
- hadde
- han
- hans
- har
- hennar
- henne
- hennes
- her
- hit
- hjå
- ho
- hoe
- honom
- hoss
- hossen
- hun
- hva
- hvem
- hver
- hvilke
- hvilken
- hvis
- hvor
- hvordan
- hvorfor
- i
- ikke
- ikkje
- ingen
- ingi
- inkje
- inn
- innen
- inni

- ja
- jeg
- kan
- kom
- korleis
- korso
- kun
- kunne
- kva
- kvar
- kvarhelst
- kven
- kvi
- kvifor
- lage
- lang
- lik
- like
- man
- mange
- me
- med
- medan
- meg
- meget
- mellom
- men
- mens
- mer
- mest
- mi
- min
- mine
- mitt
- mot
- mye
- mykje
- må
- måte
- ned
- nei
- no
- noe
- noen
- noka
- noko
- nokon
- nokor
- nokre
- ny
- nå
- når
- og
- også
- om
- opp
- oss
- over
- på
- rett
- riktig
- samme
- seg
- selv
- si
- sia
- sidan
- siden
- sin
- sine
- sist
- sitt
- sjøl
- skal
- skulle
- slik
- slutt
- so
- som
- somme
- somt
- start
- stille
- så
- sånn
- tid
- til
- tilbake
- um
- under
- upp
- ut
- uten
- var
- vart
- varte
- ved
- verdi
- vere
- verte
- vi
- vil
- ville
- vite
- vore
- vors
- vort
- være
- vært
- vår
- å

Open-source list two (https://www.ranks.nl/stopwords/norwegian):

- alle
- at
- av
- bare
- begge
- ble
- blei
- bli
- blir
- blitt
- både
- båe
- da
- de
- deg
- dei
- deim
- deira
- deires
- dem
- den
- denne
- der
- dere
- deres
- det
- dette

- di
- din
- disse
- ditt
- du
- dykk
- dykkar
- då
- eg
- ein
- eit
- eitt
- eller
- elles
- en
- enn
- er
- et
- ett
- etter
- for
- fordi
- fra
- før
- ha
- hadde
- han
- hans
- har
- hennar
- henne
- hennes
- her
- hjå
- ho
- hoe
- honom
- hoss
- hossen
- hun
- hva
- hvem
- hver
- hvilke
- hvilken
- hvis
- hvor
- hvordan
- hvorfor
- i
- ikke
- ikkje
- ingen
- ingi
- inkje
- inn
- inni
- ja
- jeg
- kan
- kom
- korleis
- korso
- kun
- kunne
- kva
- kvar
- kvarhelst
- kven
- kvi
- kvifor
- man
- mange
- me
- med
- medan
- meg
- meget
- mellom
- men
- mi
- min
- mine
- mitt
- mot
- mykje
- ned
- no
- noe
- noen
- noka
- noko
- nokon
- nokor
- nokre
- nå
- når
- og
- også
- om
- opp
- oss
- over
- på
- samme
- seg
- selv
- si
- sia
- sidan
- siden
- sin
- sine
- sitt
- sjøl
- skal
- skulle
- slik
- so
- som
- somme
- somt
- så
- sånn
- til
- um
- upp
- ut
- uten
- var
- vart
- varte
- ved
- vere
- verte
- vi
- vil
- ville
- vore
- vors
- vort
- være
- vært
- vår

- å

Open-source list three (https://github.com/xiamx/node-nltk-stopwords/blob/master/data/stopwords/norwegian):

- alle
- at
- av
- bare
- begge
- ble
- blei
- bli
- blir
- blitt
- både
- båe
- da
- de
- deg
- dei
- deim
- deira
- deires
- dem
- den
- denne
- der
- dere
- deres
- det
- dette
- di
- din
- disse
- ditt
- du
- dykk
- dykkar
- då
- eg
- ein
- eit
- eitt
- eller
- elles
- en
- enn
- er
- et
- ett
- etter
- for
- fordi
- fra
- før
- ha
- hadde
- han
- hans
- har
- hennar
- henne
- hennes
- her
- hjå
- ho
- hoe
- honom
- hoss
- hossen
- hun
- hva
- hvem
- hver
- hvilke
- hvilken
- hvis
- hvor
- hvordan
- hvorfor
- i
- ikke
- ikkje
- ikkje
- ingen
- ingi
- inkje
- inn
- inni
- ja
- jeg
- kan
- kom
- korleis
- korso
- kun
- kunne
- kva
- kvar
- kvarhelst
- kven
- kvi
- kvifor
- man
- mange
- me
- med
- medan
- meg
- meget
- mellom
- men
- mi
- min
- mine
- mitt
- mot
- mykje
- ned
- no
- noe
- noen
- noka
- noko
- nokon
- nokor
- nokre
- nå
- når
- og
- også
- om
- opp
- oss
- over
- på

- samme
- seg
- selv
- si
- si
- sia
- sidan
- siden
- sin
- sine
- sitt
- sjøl
- skal
- skulle
- slik
- so
- som
- som
- somme
- somt
- så
- sånn
- til
- um
- upp
- ut
- uten
- var
- vart
- varte
- ved
- vere
- verte
- vi
- vil
- ville
- vore
- vors
- vort
- være
- være
- vært
- vår
- å

Appendix B

Exam questions

Exam 2018

1. Variabler deklarert med var og let

Forklar kort hvilket scope som gjelder for variabler som er deklarert med nøkkelordet **var** og hvilket scope gjelder for variabler deklarert med nøkkelordet **let**?

2. Arrow-funksjoner

Forklar kort hvordan arrow-funksjoner skiller seg fra vanlige funksjoner i Javascript, med tanke på **this**

3. CSS-grid og Flexbox

Forklar kort hva CSS-grid er og CSS-flexbox er. Beskriv hvilket problem/behov de løser og hva som skiller disse to løsningene.

4. SVG og HTML5 Canvas

SVG og HTML5 Canvas kan begge brukes til å lage interaktiv grafikk på websider. Forklar kort hva begge er og gi et eksempel på (og argument) for en anvendelse hvor SVG er godt egnet og en anvendelse hvor HTML5 Canvas er godt egnet.

5. jQuery

Hva er selector-mekanismen i jQuery. Gi et eksempel og en kort forklaring.

6. Single Page Application

Hva kjennetegner en SPA (Single Page Application)?

7. Responsiv webdesign

Hva er responsiv web design? Nevn forskjellige teknikker som brukes for å implementere responsiv webdesign.

8. React (dropped)

Lag en React-komponent kalt HelloWorld for et H1 element med teksten "Hello World!".

Komponenten skal ha følgende import-statement

import React, { Component } from 'react';

og skal kunne importeres av andre javascript-filer.

9. React props og state

Forklar kort hva props og state er i React.

10. React dataflyt

Forklar hvordan du må implementere dataflyt oppover i et React komponenthierarki.

11. Web storage

Hvilken funksjonalitet tilbys gjennom HTML5 Web storage api'et (og det tilsvarende AsyncStorage api'et i React native)?

12. React vs. React native

Beskriv hva som typisk kan gjenbrukes og hva som typisk ikke kan gjenbrukes hvis du skal gjøre om en React for web applikasjon til React native.

13. State management

Hva er og hvorfor bruker vi state management som Redux og Mobx? Gi eksempel på hvordan disse brukes i implementasjonen (dvs. vis litt kode).

14. Snapshot-testing

Forklar hva snapshot-testing er?

15. REST/GraphQL

Forklar kort hva REST er eller hva GraphQL er (velg en av disse). Vis eksempel.

Exam 2019

1. Responsiv web-design

1. Hva er responsiv web design?
2. Nevn forskjellige teknikker som brukes for å implementere responsiv webdesign.

2. Interaktiv grafikk

Hvordan implementeres interaktivitet på "figurnivå" i henholdsvis SVG/DOM og med Canvas API'et?

Med interaktivitet på figurnivå menes at bruker skal kunne klikke på enkeltelementer og utføre handliger på disse (f.eks. flytte en sirkel, endre farge på et rektangel etc).

3. REST vs. GraphQL

REST og GraphQL er to forskjellige løsninger for klient-server kommunikasjon i webapplikasjoner. Beskriv og diskuter disse kort (legg vekt på å sammenligne).

4. Testing

Forklar kort hva følgende former for testing er (og hva de brukes til å teste).

- Cross-browser testing
- Enhetstesting
- Snapshot-testing
- End-to-end testing

5. Kodeforståelse I

Hva er de to viktige forskjellene mellom en variabel som er deklarert med **const** i Javascript ES6 og en variabel som er deklarert med **var** (pre ES6)?

6. Kodeforståelse II

```
const groupBy = key => array =>
    array.reduce((objByKeyValue, obj) => {
        const value = obj[key];
        objByKeyValue[value] = (objByKeyValue[value] || []).concat(obj);
        return objByKeyValue;
    }, {});
    const cars = [
            { brand: 'Audi', color: 'black' },
            { brand: 'Audi', color: 'white' },
            { brand: 'Ferarri', color: 'red' },
            { brand: 'Ford', color: 'white' },
            { brand: 'Peugot', color: 'white' }
    ];
```

Forklar hva funksjonen **groupBy** gjør og gi et eksempel på bruk gitt cars-variabelen? Vi er ute etter overordnet funksjonalitet og bruk -- og ikke detaljene i kallet til reduce.

7. Design av søkeapplikasjon

Du skal være med å utvikle en søkeapplikasjon for en samling av vitenskapelige artikler. Det skal være mulig å søke på forfatter, emne, tidsskrift, år, tittelord og databasen inneholder omtrent 1 million artikler. Det skal være mulig å filtrere og sortere resultatsettet som returneres fra et søk.

Lag en punktliste med opp til 5 gode råd for design og arkitekturen til systemet og argumenter etterpå kort for hvorfor disse er viktige.

8. Global state management

Hva er hensikten og fordelene med global state management i React applikasjoner?

NTNU

Norwegian University of
Science and Technology