Master's thesis

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

Annabelle Solem Almås

The effect of noise removal to support automatic grading of natural language

Master's thesis in Computer Science
Supervisor: Trond Aalberg
June 2022

NTNU
Norwegian University of
Science and Technology

Annabelle Solem Almås

The effect of noise removal to support automatic grading of natural language

Master's thesis in Computer Science
Supervisor: Trond Aalberg
June 2022

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

The most common way for the educational system to evaluate their students' knowledge is with the use of learner tasks and exams. These tests usually take a long time to grade, which results in less time the teacher can use to teach their students. By developing an automatic grading system that can accelerate the grading process, the freed-up time teachers now have can be used towards further assisting their students. However, the process of developing automatic grading systems is time-consuming and full of challenges. One of the challenges is to convert the student's answers (which are full of noise, for example spelling errors and language variations) into a language a machine can understand with as little noise as possible. This thesis will investigate the effect noise removal has on automatic grading for Norwegian short answers within the field of Computer Science.

This thesis uses two exams from the field of Computer Science to analyze textual noise. Most of the answers are written in Norwegian Bokmål, but some are written in Norwegian Nynorsk and English. The dataset is full of technical terms, both in English and Norwegian, which makes it complex and noisier than clean English answers, where most studies have been done.

To get familiar with the categories of noise that can be expected in exam answers, a chapter has been dedicated to look at this. Based on the identified noise categories, a deeper noise analysis was conducted for 7 of them. These noise categories are: distracting words, spelling variation, spelling errors, inflection, distracting characters, wanted terminology, and synonyms.

Two experiments were conducted to determine the impact the noise has on these exam texts. A similarity score was calculated between reference answers and student answers to figure this out. The first experiment focuses on how significant the effect that a "perfect" noise removal pipeline can contribute to the comparison algorithm's final result. The "perfect" noise removal performed on average 60% better than the unprocessed answer. The second experiment focuses on which noise category improvements impacted the results the most. 5 tests with different noise categories were tried. The tests concerning distracting words and word inflection showed the most impact. In the test where distracting words were not removed, the pipeline performed 28.04%-33.05% worse. In the test where inflection was normalized, the pipeline performed 4.09%-39.86% worse. The spelling variation test also impacted the answers that contained a lot of language variations.

The value gained from this study shows that the research into Norwegian natural language processing still has some way to go before it can compete with its English counterpart. It is challenging to develop any working system with the current resources available. Therefore, the next step needed within this field is to develop good libraries for synonym usage, term usage, and methods for handling texts with multiple languages, all publicly available.

Sammendrag

Den vanligste måten utdanningssystemet får evaluert studentenes kunnskap på er ved bruk av elevoppgaver og eksamener. Det tar ofte lang tid å sette karakter på oppgaver. Dette fører til at lærere har mindre tid til faktisk å lære studentene. Ved å utvikle et system for automatisk karaktersetting kan prosessen for karaktersettingen bli kortere, som igjen kan frigi tid som kan brukes til å hjelpe studentene. Prosessen å utvikle automatiske karaktersettingssystemer er imidlertid tidkrevende og full av utfordringer. En av utfordringene er å konvertere elevens svar (som er fulle av støy, for eksempel stavefeil og språkvariasjoner) til et språk en maskin kan forstå med minst mulig støy. Denne oppgaven skal undersøke hvilken effekt støyfjerning har på automatisk karaktersetting for norske kortsvar innen fagområdet datavitenskap.

Denne oppgaven bruker to eksamener fra fagområdet datavitenskap for å analysere tekstuell støy. De fleste av besvarelsene er skrevet på norsk bokmål, men noen er skrevet på nynorsk og engelsk. Datasettet er fullt av faguttrykk, både på engelsk og norsk, som gjør at besvarelsene inneholder mer støy og er komplekse å rense. De inneholder dermed en del mer støy enn rene engelske besvarelser, som de fleste studier er gjennomført på.

For å sette seg inn i hvilke støykategorier som kan forventes i eksamensbesvarelser, er det viet et helt kapittel for å se på dette. Basert på de identifiserte støykategoriene ble det utført en dypere støyanalyse for 7 av dem. Disse støykategoriene er: distracting words, spelling variation, spelling errors, inflection, distracting characters, wanted terminology og synonyms.

To eksperimenter ble utført for å finne ut hvilken innvirkning støyen har på eksamenstekster. Det ble beregnet en likhetsscore mellom referansesvar og elevsvar for å finne ut av dette. Det første eksperimentet fokuserte på hvor betydelig effekt en ”perfekt” støyfjerningspipeline kan bidra til sammenligningsalgoritmens endelige resultat. Den ”perfekte” støyfjerningen presterte i gjennomsnitt 60% bedre enn det ubehandlede svaret. Det andre eksperimentet fokuserte på hvilke støykategorier som påvirket resultatet mest. 5 tester med ulike støykategorier ble testet. Testene med distracting words og inflection viste mest effekt. I testen der distracting words ikke ble fjernet, presterte pipelinen 28,04%-33,05% dårligere. I testen hvor ord ble normalisert, presterte pipelinen 4,09%-39,86% dårligere. Spelling variation testen påvirket også svarene som inneholdt mange språk variasjoner.

Verdien hentet fra denne studien viser at forskning på norsk naturlig språkbehandling fortsatt har et stykke igjen før den kan konkurere med sin engelske motpart. Det er vanskelig å utvikle et fungerende system med dagens ressurser. Derfor er neste steg som trengs innenfor dette feltet å utvikle gode biblioteker for synonymbruk, faguttrykk og metoder for å håndtere tekster med flere språk, alt offentlig tilgjengelig.

Table of Contents

List of Figures

List of Tables

1 Introduction

Picture a future where feedback and grading of learning tasks and exams using natural language are given within less than an hour or perhaps seconds from delivery. This can be done by utilizing automatic grading systems. At NTNU today, the grading deadlines are 3 weeks. However, this is often extended due to holidays or sick leave [26]. Grades and feedback are given so long after the test that it is difficult to learn anything from them. Educational systems will benefit by implementing precise automatic grading systems. The time teachers spend grading can instead be used towards helping students or improving lectures. Automatic grading can also contribute to more objective and fair grading. One of the problems with using humans to grade is that they are not always consistent. The personal relationship between students and teachers can be a factor, even when the grading is done anonymously, the choice of language used can affect the grading. The time of day the teacher grades, their mood, fatigue, or the use of multiple teachers can also contribute to different grades for equally correct answers.

Research within this field dates back to 1966 with the system Project Essay Grader by Ellis Page [28]. However, few NLP grading systems are in use today. This is not to say that they do not exist in the educational system, but the systems in place are not impressive, nor do they resolve the problems that arise with human grading. This is the case because of the complex and sometimes near to impossible task of developing and maintaining these systems. There are many challenges within this field. In the greater picture, they can be categorized into 5 challenges. Generating good questions for the system, developing good reference answers to match the answers against, preprocessing the answers correctly to help the computer match, developing precise grading algorithms, and methods for giving good feedback.

This thesis will focus on the challenge of text preprocessing. For a system to come to the step of grading the answer, it needs first to be processed. Unfortunately, a human exam answer often contains a lot of ”noise” that the computer has problems understanding and comparing. The noise can be anything from different word inflections and spelling errors to the use of synonyms. Removing noise is a complex task, and the noise differs for each question set. Therefore, this study will research the textual effect noise has on the performance of an automatic grading system. The study will use exams from the field of Computer Science written in Norwegian. With this in mind, the following research questions have been created.

RQ 1: Which are the different noise categories found in Norwegian exams?

RQ 2: Does preprocessing, through noise removal, contribute to a better comparison of Norwegian Computer Science exam answers?

RQ 3: Which parts of a preprocessing pipeline has the greatest impact?

1.1 Contributions

This thesis contributes to the research area by getting a better understanding of how important noise reduction can be for automatic grading with a focus on Norwegian exams. Most studies on automatic grading focus on the results in the actual grading algorithm and focus less on the value good noise reduction in the answers contributes to. There is an understanding that noise reduction is important, but little study on how important this is. Therefore, this thesis' contributions are:

1. An overview of existing automatic grading systems and their relationship to text preprocessing/noise reduction.
2. An overview of different noise sources often present in exam answers.
3. An analysis of noise sources present in the dataset studied (Norwegian Computer Science short answers), to evaluate their impact on the texts.
4. An experimental study of the effect noise removal has on a grading result and which of them that affects the grading results the most.
5. A proposal of further work needed within this area.

Outline

Chapter 2: Overview of concepts and theories that are used in this thesis. This chapter is a basic introduction to natural language processing (NLP) created for the reader with no prior knowledge of this subject area.

Chapter 3: Presents the concept automatic grading system. It describes the importance of defining text categories before developing a system and explain some automatic grading systems used through time.

Chapter 4: Presents different types of noise in natural language, where they often occur, and methods to handle them.

Chapter 5: An analysis of the dataset with a focus on the amount of noise that exists in the dataset.

Chapter 6: In this thesis, two experiments were conducted. Experiment 1 focuses on the best possible effect noise reduction could have on the results. Experiment 2 focuses on the effect the different noise sources have on the result.

Chapter 7: Discusses the thesis findings.

Chapter 8: Concludes the thesis, and presents some possible options for further work within the area of noise reduction.

2 Theory

The goal of the theory chapter is to introduce the concepts used in the thesis. It is a simple introduction to Information Retrieval and Natural Language Processing. The chapter will introduce knowledge needed for a reader with no prior knowledge of the topic to follow the thesis effortlessly.

2.1 Information retrieval

Information retrieval (IR), with regards to natural language, is the process of finding unstructured text (referred to as documents) which satisfies a set of search criteria (referred to as a query) and then ranking these documents by relevance. IR is commonly used to find relevant documents on Google, news articles in online newspapers, scientific articles on Google Scholar, and more. As presented in figure 1, the IR pipeline consists of a set of documents, a query, and - between them - an IR model used for indexing, searching, and ranking. The output of the model is a ranked set of documents based on the query. Automatic Grading Systems utilize IR methods to both process answers as well as querying whether an answer is a good match against the reference answer.

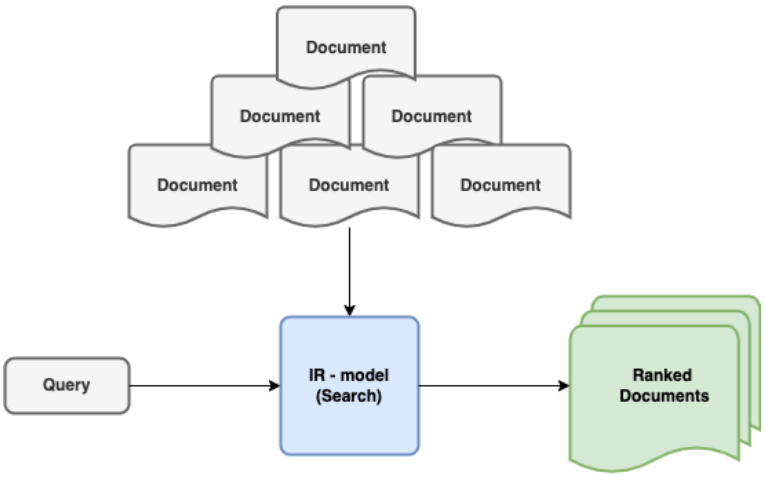


Figure 1: A simplified presentation of information retrieval.

2.2 Text representation - natural language processing

For a computer to be able to analyze and compare natural text, the text has to be converted to a language a computer can understand. Computers work best with something they can mathematically evaluate, so the best representation for the text is a numerical one. This is often referred to as natural language processing (NLP). One such representation can be a multidimensional vector, where the tokens (elements the text consists of) are represented as different numbers in the vector.

2.2.1 N-grams

N-grams are the concept of dividing sentences into tokens of varying lengths, as presented in figure 2. They come in many different forms, such as uni-grams, bi-grams, tri-grams, and four-grams. N-grams bigger than one are used to find neighbor relations and contextual relations between words [20].

The	effect	of	noise	removal	to	support	automatic	grading

Uni-grams

The effect	effect of	of noise	noise removal	removal to	to support	support automatic	automatic grading

Bi-grams

The effect of	effect of noise	of noise removal	noise removal to	removal to support	to support automatic	support automatic grading

Tri-grams

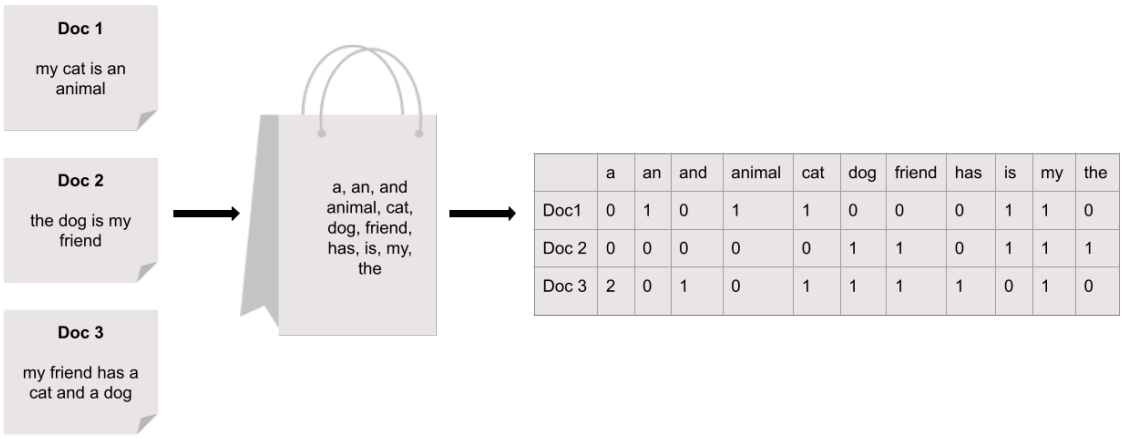
Figure 2: Presents the uni-grams, bi-grams, and tri-grams for the sentence ”The effect of noise removal to support automatic grading”.

2.2.2 Corpus

A corpus is a collection of documents (texts) used with one defined purpose. The corpus is often used as a resource in learning models to define a set of characteristics to be used within a text category. For example, it can be a text collection that focus on meaning (specific topic), the writing form (correct grammar or finding normalization), or language translations. When referring to multiple corpus it is called corpora. When requiring a corpus to do a task, often corpora is needed. For example, when grading exams, one unique corpus is needed for each question. [46]

2.2.3 Bag-of-Words

Bag-of-words is one of the simplest methods for representing text in NLP and IR. Word frequency is the main feature, where word order and grammar is not important. All tokens in a corpus are put into a ”bag”, and text vectors are generated from the frequency of words in each text. This is illustrated in Figure 3. A token can also use other weights than word frequency, which allows a text to calculate vectors with words of different importance.



	a	an	and	animal	cat	dog	friend	has	is	my	the
Doc1	0	1	0	1	1	0	0	0	1	1	0
Doc 2	0	0	0	0	0	1	1	0	1	1	1
Doc 3	2	0	1	0	1	1	1	1	0	1	0

Figure 3: Bag-of-words presented with Term Frequency.

2.2.4 Weigthing method

There are multiple methods to decide the weight of a token in a corpus. Some weighting methods are Term Frequency (**TF**) and Term Frequency-Inverse Document Frequency (**TF-IDF**)

Term Frequency is the number of times a word occurs in a document. The more a word is used the higher the weighing of the word.

Term Frequency Inverse Document Frequency consist of Term Frequency as mentioned above, and Inverse Document Frequency (IDF). IDF is the measure of how many documents a token occurs in. It is used to find out how important a token is. If a token occurs often in one document it might be an indication that the token is important for that document, but if the token also occurs in a lot of other document it might indicate that the token is not important. To get the TF-IDF the TF and the IDF are multiplied.

2.3 Cosine similarity

Cosine similarity is a measure of how similar two texts are to one another, regardless of their lengths. Cosine similarity is beneficial because while one text might only be one sentence and the other an entire paragraph, they can still be similar in content even though they are dissimilar in length. The measurement is done by calculating the cosine angle between the two vectors representing each text. If the angle is small, they are closer together and therefore similar. For example, figure 4 shows the cosine similarity between the phrases ”Hi, world!” and ”Hello, world!”. Due to the lack of dimensions, these two vectors are pretty dissimilar, but if considering a text with hundreds of dimensions, a one-word difference would not create such a wide angle.

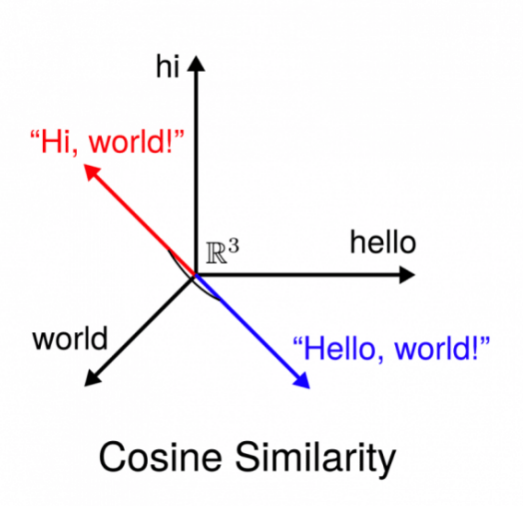


Figure 4: Cosine Similarity between the phrases ”Hi, world!” and ”Hello, world!” [5].

2.4 Levenshtein distance

Levenshtein distance, also known under the name edit distance, is a way of quantifying how dissimilar two strings are to one another, by counting the minimum number of single-character operations required to transform one string into the other [50]. These operations are insertion, deletion, and substitution. In natural language processing, it is often used to determine candidate words to substitute a misspelled word. For example, ”Hollow” has a Levenshtein distance of 2 to ”Hello”. This is because the first ”o” needs to become an ”e”, and the ”w” needs to be deleted.

H	O	L	L	O	W
	s				**d**
H	E	L	L	O	

Table 1: The Levenshtein edit steps from ”Hollow” to ”Hello”. The **s** symbolizes substitution and the **d** symbolizes deletion.

3 Related Work

Automatic grading systems are computer programs developed to assist grading. Creating an automatic grading system can be divided into 5 steps. 1) Data creation, where questions that can easily be compared to a reference answer are created. 2) Create reference answers or a reference model to compare the student answers against. This may be one or multiple answers. 3) Create a text preprocessing pipeline to remove noise from the student answers and convert the answers into a language a computer can understand. 4) Create a grading model to compare and grade the student answers against the reference answers. 5) Data resolution, evaluating the grading model and give feedback to the user. The model presented in figure 5 shows the process pipeline of automatic grading.

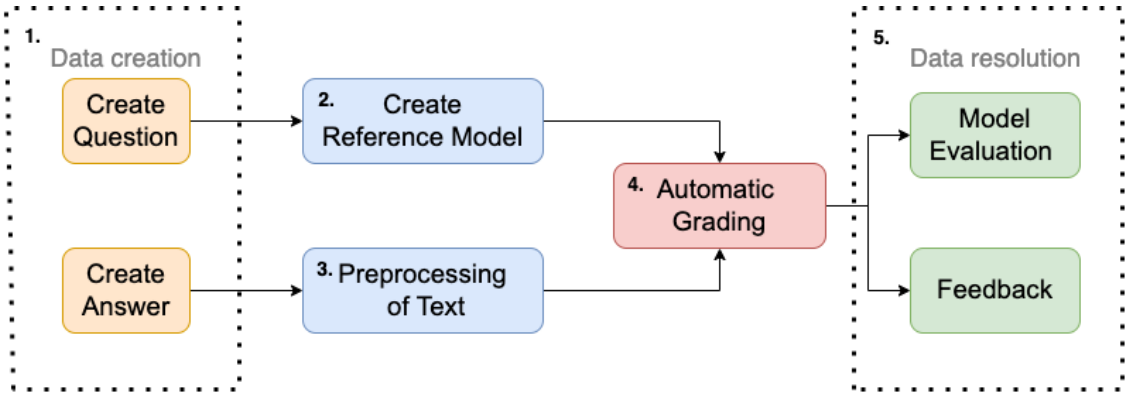


Figure 5: A simplified process pipeline for an automatic grading system.

To better understand automatic grading systems, this chapter will present the different categories of text a system can be developed for and some studies and systems already created.

3.1 Text categories

Text category identification is the first step in developing an automatic grading system. The system is dependent on the type of text to get precise results. Three main categories of identification are whether or not the text is a Short Answer or an Essay Answer, if the text is Structured or Unstructured, and if the question to answer is Closed- or Open-ended. In this section, these categories are presented.

3.1.1 Short Answer vs Essay Grading

Before beginning the process of developing an automatic grading system that evaluates natural languages, it is essential to decide on the answer length the system is supposed to handle. The most common categories of texts, with respect to length, are Essays and Short Answers (SA).

Essays are long texts that include multiple details, concepts, topics, and statements. They can span anywhere from a couple of paragraphs to several pages [4]. The extensive variance between Essays makes them difficult to compare to one another or evaluate.

Short Answers are typically limited to, at most, a single paragraph in length and is only supposed to cover a single topic in brevity. Usually, one can expect from 1-4 sentences up to a paragraph, with 15-20 words per sentence for plain English [13, 34], but academic or scientific texts trend towards 25-30 words per sentence [45], so there can be differences here as well. While Short Answers, due to their nature, offer less chance for variance than their longer counterparts, there is still room for uniqueness. According to Grammarly [11], a paragraph should contain between 100 and 200 words. From this, we can derive that a Short Answer should be between 20 and 200 words, while Essay answers should contain 200 words or more. Due to these potentially vast differences in length, the automatic grading process for each is usually split into 2 aptly named categories: Automatic Essay Grading (AEG) and Automatic Short Answer Grading (ASAG).

Automatic Essay Grading (AEG) focuses more on the structure of the text [22], grammar and sentiment are essential elements for the final grading, as well as the content. Automatic Short Answer Grading (ASAG) focuses on the content of the text, where the structure is less significant [4, 14]. For short answers, there should always be a clearly defined correct answer [22]. It is stated by the authors of C-rater (an ASAG system) that the goal is to "map student responses onto the experts' models in order to determine their correctness or adequacy", while for E-rater (an AEG system), the goal is to "focus on metrics that broadly correlate with the writing style, augmented with aggregate measures of vocabulary use" [4]. The research by Klein et al. [14] focuses on short answer grading, analyzes texts only based on content, but states that including style as a feature will be the next step.

3.1.2 Structured vs Unstructured

There are two types of text: structured and unstructured [4]. Structured texts come in well-defined formats, such as source code for programs or mathematical equations. Unstructured texts, on the hand, are more complex, and it's near on impossible to predict what they will look like before they've been written. This is due to the unknown form of natural languages. Where source code and maths can be parsed and evaluated syntactically, sentences in natural languages can be written in so many ways that it renders parsing them difficult. Thus the processing of both varies in both complexity and time consumption, with the more demanding of the two being unstructured text. Mathematical equations such as $x^2 + y^2 = z^2$ and source code such as `print('Hello world!')` are both structured text forms. Most sentences in natural languages are unstructured forms of text.

3.1.3 Closed- vs Open-ended questions

The complexity of an automatic grading system is determined by the question type. Questions can be categorized into two primary groups—closed-ended and open-ended questions. A closed-ended question requires the respondent to recognize or identify some key information. This can be yes or no questions, multiple-choice, rank order questions, or fill in the missing word [22, 4, 43]. This question type requires minimal implementation because the learner's answer can only be correct or incorrect. Therefore, the grading is easy, and feedback to learners can be quick and accurate [43]. An open-ended question requires the respondent to come up with a unique answer by writing. This is often in the form of written natural language text such as Short Answers or Essays but can also be in the form of a code language, like programming snippets. The list of semantically equivalent answers can appear to be almost limitless. It requires a complex grading system that manages to compare different texts as equally correct. This is not an easy task and requires a complex prepossessing of the texts, as well as good reference models [4, 43, 18]. A question can also be categorized as in between closed-ended and open-ended. This is where the question can be written in multiple ways, but the semantically equivalent correct answers are very similar.

3.2 Different automatic grading systems

According to Burrows et al. [4], automatic grading of short answers has had a natural progression in methods used throughout time. In the earlier systems, concept mapping and information extraction methods were the dominant techniques. Later, corpus-based methods and machine learning have become more prominent. These systems can be categorized as either Rule-based or Statistical, and are presented in figure 6 [4].

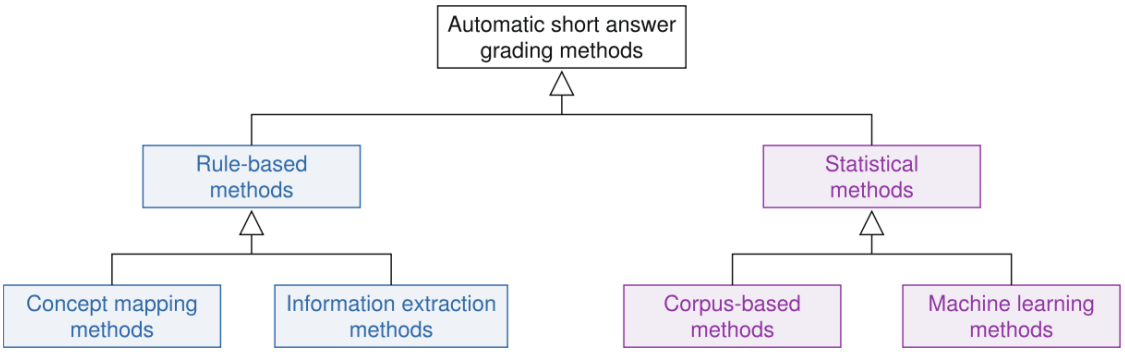


Figure 6: Rule-based and Statistical methods [4].

3.2.1 Concept mapping

Concept mapping revolves around the idea that every sentence (in this case, answers) is made up of one or more concepts. An algorithm will try to identify as many of these concepts as possible and then compare them with the concepts extracted from one or more reference answers. At least one such reference answer is needed for every question. There are several ways to identify concepts. One way could be to extract a single concept from a sentence, but one could also make an even more nuanced comparison and extract a concept from combinations of 2 to 3 words [4].

To better understand how concept mapping works, let's take a look at a system that has successfully used this method for Automatic Short Answer Grading.

3.2.1.1 The system C-rater

Concept Rater [18], better known as C-rater from 2003 tries to match all possible concepts between respondent and reference answer with the use of specified sentence level concepts that are extracted from each answer. The system used a model that a content expert manually created for each exam. This tailoring is done to map the student's answer to the reference answers in the most beneficial way. The system includes a graphical user interface to generate reference answers. This is done to ease the complex process of creating reference answers that cover all concepts. It works by breaking down an answer sentence into simple concepts and adding a set of sentences that satisfies one or more of these concepts. For each sentence, a tuple model is generated. For each tuple model, the user can select which words are essential and create a list of synonyms/antonyms. After this process has been completed for every question, C-rater is ready to evaluate the student's answers for that specific set of answers [4, 18, 35].

Concepts can be expressed in an exhaustive amount of ways within the natural language. C-rater uses a "paraphrase recognizer" to identify the presence or absents of concepts represented in the text. C-rater normalizes/preprocesses a sentence based on four sources of variation: syntactic, pronoun references, morphological, and synonym use [18]. These can all be categorized as noise.

Syntactics To accurately compare the meaning of different sentences, C-rater converts each sentence into a canonical tuple form with the sentence's verb taking the subjects/objects as arguments. It then normalizes each of the tuples instead of the natural sentences. This allows C-rater to distinguish between sentences that might have significant visual similarity, all the while being syntactically dissimilar [18].

Pronoun references To deal with the ambiguity of pronoun association in sentences, C-rater makes use of a technique developed by Morton 2000 [23] where they substitute pronouns with the subject/object. This technique has been trained on student responses to essays and short answer questions. An example of this technique could be the sentence ”Take one plant and set it in a dark closet”. In this case, ”it” would be replaced with ”one plant” [18], leading to the sentence ”Take one plant and set one plant in a dark closet”. This sentence might seem weird to humans, but it conveys the writer's intention much more clearly to a machine.

Morphological Each word is normalized to its base form to avoid comparing several variations/conjugations of the same word. Examples of this are the words ”subtracts”, ”subtracting”, and ”subtracted”. These are all conjugations of the word ”subtract”. This also allows matching the verb ”subtracts” with the noun ”subtraction”. Another important facet is the concept of negating sentences. By stripping negating prefixes, such as un-, from the words, and adding a ”not” identifier to the tuple form, C-rater can store positive/negative intent accurately.

Synonyms Since a student's vocabulary and a teacher's vocabulary might differ from each other [18], it can cause the results to have lower accuracy than they should. By utilizing a ”word similarity matrix”, with synonyms developed from over 300 million English words, C-rater can remove the dissimilarity by finding synonyms and similar writing forms for all the tuples and replace some of them to match the rest.

Concept matching Comparing student answers to their reference model counterpart is done by following a set of rules. One of the rules mentioned is that ”subjects and objects cannot be interchanged (except for a small class of verbs)” [18]. This is done to avoid the case where the sentence ”a dog bit a man” is equivalent to ”a man bit a dog”. A bag-of-words approach to this problem would result in the two sentences being the same, as they contain the exact same words. This is, for obvious reasons, not desired.

Typical errors Of the many errors that can arise while grading student answers, C-rater is mainly concerned with ”misses” and ”false positives”. A ”miss” would be when an answer is not given its deserved credit while providing a correct answer. An example of this could be writing ”fire alarms take a chunk of change” instead of the reference answer ”fire alarms are expensive”. ”False positives”, on the other hand, are when an answer is marked as correct while not accurately answering the questions. Setting aside objectively incorrect answers, most of these errors arise when ”a student does not know when to stop”. Starting by writing a correct answer, and then adding on more incorrect information is an example of this. C-rater tries its best to look at an answer in the best way possible. Thus, it has to allow these ”fragmentary responses” for better matching [18].

3.2.2 The value of concept mapping

Concept mapping has shown to be a system structure that is demanding to construct. It is not applicable for questions in fields of rapid change, such as Computer Science exams, due to the time-consuming task of tailoring the system to each new question set. The methods used to process the text, on the other hand, are quite useful and can easily be applied to other automatic grading techniques. The benefit of using C-rater is that it is well constructed to identify all possible ways of writing the same text. Normalizing text and minimizing the use of synonyms can enhance the possibility of finding similarities with the reference answers. Pronouns substitution can lead to a better comprehension because the machine now knows what all of the elements in the text refer to. Incorporating all of these elements in a general preprocessing pipeline will contribute to a lot of noise removal.

3.3 Information extraction

Information extraction revolves around the idea of ”fact finding”. These are facts that can be searched for effortlessly. It can be defined as ”a series of pattern matching operations” [4]. Regular expressions and parse trees are two of the methods for pattern matching [4]. The technique can be used to extract unstructured content based on structured data, where the structured data are represented as tuples. Information extraction can only be used if the question is specific enough, and the criteria for correct answers are clearly defined [41].

Studies argue that information extraction is relatively robust for texts with incomplete and incorrect grammar in the answer and is an easy method to implement [41]. To better understand how information extraction works, let's take a look at two systems that have successfully used this method for Automatic Short Answer Grading.

3.3.1 The system Auto-marking

Auto-marking from 2003 is a method developed to fit a set of hand-crafted patterns to a training model on answers within biology [4, 41]. The research included a lot of manual labor to develop good patterns. First, the system's authors created reference answer material for each question. Then the reference answers were used to develop a set of extraction rules, to cover all plausible variants of answers that could be written [41]. Figure 7 presents the possible reference answers and the pattern making the system does.

```
the egg was fertilised it split in two

one egg fertilised which split into two

the fertilised egg has divided into two

1 fertilised egg splits into two
```

These all imply *It is the same fertilised egg/embryo*, and variants of what is written above could be captured by a pattern like:

```
singular_det + <fertilised egg> +
  {<split>; <divide>; <break>} + {in, into} + <two_halves>

singular_det     = {the, one, 1, a, an}
<fertilised egg> = NP with the content of 'fertilised egg'
<split>          = {split, splits, splitting, has split, etc.}
<divide>         = {divides, which divide, has gone,
                    being broken...}
<two_halves>     = {two, 2, half, halves}
etc.
```

Figure 7: An illustration of how Auto-markings develops patterns based on the reference answers.

The research does not focus on processing the text, except ”part-of-speech” tagging and error correction. However, Auto-marking had a problem where the system did not recognize the technical terms within biology. This was due to the ”Wordbank” used, which was ”not particularly rich in biological vocabulary” [41].

3.3.2 The system Auto-Assessor

Auto-Assessor from 2011 works by matching word for word on a single-sentence text against a reference answer, and uses a bag-of-word method. The system uses a part-of-speech tagging in the preprocessing step of the pipeline and synonym check using WordNet (a lexical database of semantic relations between words [52]) to match against the reference answer. For the system to work, the answers have to adhere to a strict set of rules in order to be accurately processed. To begin with, an answer must be given as a single complete sentence with proper grammar and no spelling errors [6]. It also has to be written in the English language without any usage of slang or analogies. The reason for these strict requirements is to ensure that the results are not due to any semblance of noise in the answer, but solely due to the comparison method employed. According to the authors of the system, all of these assumptions are necessary to have support for in a well functioning system [4, 6].

3.3.3 The value of Information extraction

Information extraction techniques indicate low utility in today's automatic grading systems. Even though some of these systems show great accuracy and are often easier to implement than other methods, they are more labor-intensive. Auto-marking will probably take longer to develop than it would take a teacher to grade the answers manually. Auto-Assessor, on the other hand, seems not to use that long time to develop, but it requires the answers to be written in a specific way. This is not likely to be the case for an average student answer and will only be a viable option for questions with a very concrete answer. Even though the method itself will not work, it contains many interesting details that can be utilized.

3.4 Corpus-based techniques

Corpus-based methods revolve around the idea that statistical properties in large document corpora can be exploited to get more precise results [4]. A text preprocessing model, trained on a well-selected corpus within the topic of the questions, will manage to find and fix more noise than static dictionaries will. Models can be tailored to the answers expected by training a model utilizing texts with the correct language and regional variations, terminology, expected grammar, and spelling errors. Corpus-based techniques are often used for longer-text systems, such as Essay Answer Grading. However, it can also be used for shorter texts with regards to writing styles and synonym use for Short Answer Grading [4]. A lot of ASAG methods focus on the semantic similarity between the learner and the reference answers [22, 4, 14]. If the corpus data is accurate enough, this technique will be able to find similarities exactly.

To better understand how corpus-based techniques work, let's take a look at some studies using this method for Automatic Grading.

3.4.1 The studies by Klein et al. and Lubis et al.

"Latent semantic analysis (LSA) is the most popular corpus-based semantic similarity technique" [22]. This quote is from the study done by Lubis et al. in 2021. The two studies described below use this technique. LSA focus on the extraction of meaning from the text. The idea is that words with close meaning will "appear in similar segments of text" [22]. It can be defined as "a method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text" [17].

A study by Klein's [14] from 2011 argues that the open-ended questions are very complicated to grade, as the semantic meaning of the response is more important than any of the individual keywords. Klein's approach uses student answers from the exam to develop the reference model in the grading algorithm instead of retrieving data from other sources [4, 14]. This will ensure

that the language used by the students is an important factor in the grading process. The correct terminology the students uses is used to grade. This is beneficial because a lot of dictionaries lack subject-specific terms [6, 42]. LSA is invariant to word order, so terms are treated as individual words [14].

Lubis et al. study [22], developed a system using semantic similarity technique with word embedding from an Indonesian dataset to only be dependent on a single reference answer. Word embedding is a trained vector space with tokens. Tokens that express similar meanings are close in the vector space [51]. They used the library Gensim to train the vectors. To do this, they used 365 939 article lines from a Wikipedia corpus and obtained 98 000 word vectors [22].

These studies focus on developing good reference models by utilizing the LSA. Although they do not focus on removing noise from the answers, they use the vector space with all the token variations to calculate the high similarity between tokens with "equal meaning". This will ensure that answers which are different/noiseful but equal in meaning will not be scored differently.

3.4.1.1 The research by Süzen

One of the challenges with using corpus-based techniques is that it requires the corpus to be within the topic of the answers to grade. Evaluating semantic equivalent texts with the use of a corpus from another topic can lead to poor results [42]. Then the dictionaries developed from the corpus are not specific enough to find all of the semantic equivalence in the text. There are challenges with finding qualified available corpora for some topics. If there are no corpora, a semantic dictionary can not be developed. There are a number of reasons why this is: No one has prioritized developing it, no one has made it publicly available, the language has not enough available texts within the topic, or the topic is in constant change, and maintenance of the corpus are demanding. Süzen developed corpora and dictionaries to analyze semantic equivalence in academic texts. They built the Leicester Scientific Corpus (LSC) from abstracts from 1.673.824 English written research articles and proceeding papers with over 298 million words spanning 252 subject categories [42]. With the LSC, two dictionaries were developed, namely the Leicester Scientific Dictionary (LScD) and the Leicester Scientific Dictionary Core (LScDC) [42].

> "LScD is a list of words extracted from the texts of abstracts in the LSC. It consists of 974,238 unique normalized words."[35]
>
> "LScDC is a sub-list of the LScD. The dictionary contains 104,223 unique words. It filters out all words appearing in less than 10 documents. They were removed because the words would be too rare to yield any meaningful information about the answer, as they only appear in less than 0,01% the available documents."[35]

3.4.2 The value of Corpus-based techniques

Corpus-based techniques have been useful for short answers as the focus is more on the meaning and not on the exact text. It is a method that eases the process of developing the system. Finding enough texts within the set topic makes finding texts with equivalent semantics possible. One of the problems that often occur in natural language grading systems is that technical terminology is misinterpreted. Usually, they are classified as spelling errors and changed. Systems that understand all terminology supposed to exist in the texts will manage to be more accurate. However, to attempt this task using texts written in Norwegian Bokmål would be infeasible at best. There simply do not exist enough subject-categorized articles to develop a dictionary accurately. Given English's role as an international language, it is only natural that the amount of articles at their disposal is larger than what is available in Norwegian. It also does not help that a lot of Norwegian students write their articles and theses in English, something this very thesis is guilty of as well.

3.5 Machine learning

The idea of using machine learning revolves around lowering the human effort that goes into developing the grading system. To put it simply, the system's task is to automatically find patterns that satisfy one or more categories within a given set of answers with the use of statistical methods, without requiring much, if any, human interaction. Machine learning systems use a wide variety of features, with bag-of-words and some sort of weighting algorithm being the most popular. "TF" and "TF-IDF" are two commonly used weighting algorithms. Other useful features are n-grams, sentence counts, count of common discourse connector words, synonym sets, PoS tagging, stem and lemma matching, grammatical relations, and edit distances [4, 41].

A now widely used technique in machine learning systems is text classification [4, 13], with its advantage being automatic customization to new domains by only requiring expert knowledge from a human examiner [41]. In a 2003 study by Sukkarieh et al. [41], it is argued that text classifying systems of this type would not outperform a hand-crafted system, such as information extraction. This claim was based on their study, where a k-nearest neighbour approach failed to match its information extraction counterpart. In the nearly 20 years since then, text classification has come a long way, showing promising results in the studies by Horback et al. (2013) [13] and Burrows et al. (2015) [4].

One of the more popular subsets of machine learning is deep learning, where techniques such as Long-Short-Term-Memory (LSTM) can be employed in automated grading systems [53, 54, 12]. The main benefit of deep learning is that it is designed after the human brain and as such can handle more complex tasks. Quite a few ASAG and AEG systems have been developed using deep learning.

Still, the sheer amount of data needed - both for training and for testing - makes it a challenging, if not impossible task to complete for rapidly changing fields (such as Computer Science) where data available for training does not yet exist.

3.5.1 The value of machine learning

The main contributing value of machine learning is that it minimizes the need for human interaction when creating the model for the system. And as this data needs to be historical, it is hard to use such a system in rapidly changing fields. However, retrieving good data can give high accuracy if developed correctly.

4 Noise in Natural Language Processing

What is noise in natural language? And how should one deal with noise when it occurs in a text? These are two important questions when trying to make an automatic grading system. In natural language processing, the concept ”noise” is complex, and it is challenging to create a clear uniform definition [44]. However, there are several similar definitions. Some define the term a bit loosely, and some define it in more detail. Li et al. [19] define noise as ”any unwanted disturbances superposed upon the intended speech signal”. According to Sharou et al. [1], the term ”noise” has been used to cover both harmful and meaningful types of nonstandard content. They propose dividing the term into two groups — harmful noise and useful noise. Harmful noise is defined as ”any unwanted disturbances that are either harmful or useless or both”. If the performance of an NLP system or the intended meaning of the text is affected, the noise is harmful. If the text does not serve a purpose, it is useless. The useful noise is defined as ”wanted content that serves a purpose and carries meaning that is important for the task and/or for the performance of the NLP system”.

In this paper, ”noise” will be defined as any element in a text that will make automatic analysis more difficult. The ideal is to remove all noise in a text, but some noise removal will hurt the system's performance. For example, if the text is a cooking recipe, removing noise such as numbers will remove large amount of the meaning. Buns and pancakes include most of the same ingredients but with different amounts. If the system compares cooking recipes without numbers, they would be categorized as the same. Therefore, every system needs to analyze the corpus types the system shall handle. Not only with regards to text categorization, as mentioned in chapter 3.1, but also with regards to possible noise sources that are expected to exist before further development of the system. In this way, the system does not try to remove useful noise or noise not present in the texts. This is beneficial in two ways. First, wrong text processing can lead to creating new noise. An example is removing English suffixes on Norwegian text. The word ”Adressested” becomes ”Adressest”, and can not be compared with ”Adressesteder” which becomes ”Adressested”. Second, it will introduce unnecessary processing in preprocessing and potentially for later text comparison and grade assignments.

4.1 Written and spoken text style

The different types of noise are extensive, but not all noise types are expected to exist in all text styles. A one-size-fits-all noise removal pipeline will therefore not work. By categorizing the text style, only the correct and needed noise is removed. Text can be categorized into two primal groups: written form and spoken form. These can then be divided into even more subgroups. In this paper, the written form is defined as all text that is written with the intent of correct grammar for a written language. Text types within this category are books, research papers,

school deliveries, formal letters, contracts, etc. Spoken form is defined as all text that is written without the written language in mind. This will include writing in the dialect form but also using casing, symbols, shortening, and lengthening of words to convey the meaning in the text. Text types within this category are transcribed text (from audio), Twitter texts, text messages, songs, etc. Then we have the subgroups that are a combination of both. Where the intent is to write in a written form, but some spoken parts occur. One of these subgroups is written exams for courses where the content is a crucial part of the answer. Students writing with a time limit in mind and a goal to get the best grade will make mistakes when writing. Therefore, an NLP system needs to implement a more complex noise analysis for these text styles.

4.2 Noise categories

Noise in NLP can be divided into multiple categories. In this section, noise often found in school exams is defined with a clear description. This section is inspired by the work done by Al Sharou et al. [1]. The noise categories are presented in figure 8.

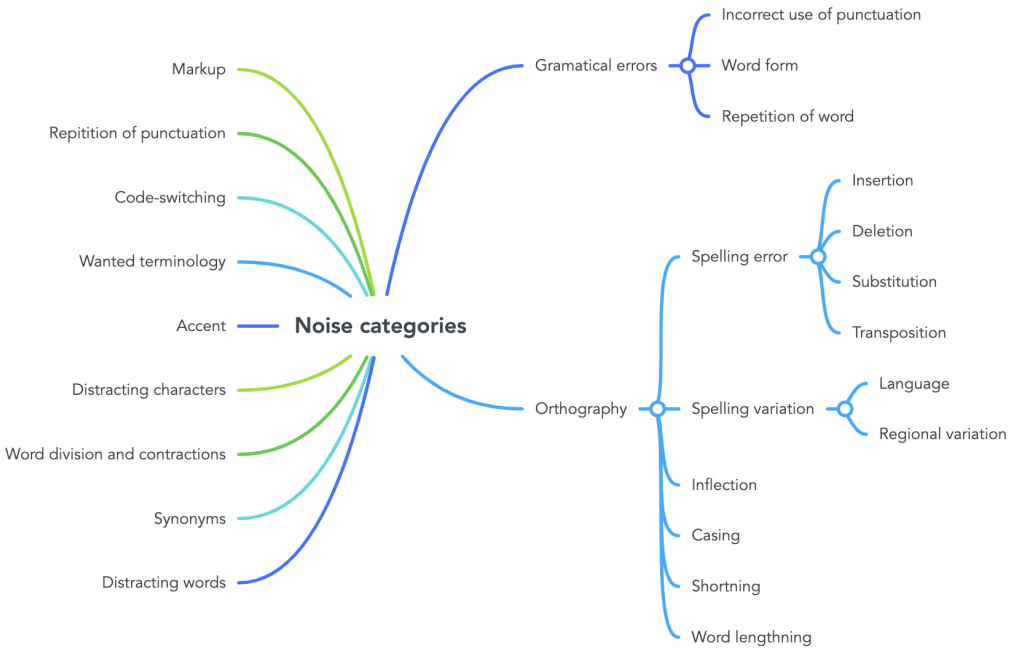


Figure 8: Noise categories within natural language processing.

4.2.1 Orthography

Orthography concerns the way words are written and uses the rules of the written language to define correct writing. This category contains noise defined as spelling variations, spelling errors, and method's for spelling variance that includes inflection, casing, shortening and word lengthening. Spelling variations are other ways of writing the same meaning, and spelling errors are words not defined in the written language [1].

4.2.2 Spelling variation

Spelling variation can be due to regional variation, where the same words can be written differently within the same language. English words can be written in the British or the American form. Norwegian words can be written in the Nynorsk or the Bokmål form. These written languages contain many of the same words, some spelled the same, some spelled differently, and some that are unique and not recognizable to the other written form. Examples of different spelling forms in English are ”colour” and ”color” or ”centre” and ”center”. In Norwegian ”helhet” and ”heilskap”, ”tillatelse” and ”løyve” or ”annet” and ”annan”. Another noise type within regional variation is that different words can mean the same thing. For example, in British English, ”football” and the American word ”soccer” mean the same thing. But ”football” in American English is another sport.

Another spelling variation can be word endings. Words can be written with different suffixes. In English a suffix can be ”ed”, ”s” or ”ize”, in Norwegian Bokmål it can be ”er”, ”ert” or ”ene”, and in Norwegian Nynorsk ”a”, ”ar” or ”ane”.

It can be important for a system to identify which spelling variation is used in a text to be able to remove the correct noise. In Norwegian school exams, this is needed because both Nynorsk and Bokmål forms are accepted. Removing noise from a Norwegian Bokmål text using the Nynorsk language rules can make a text even noisier than before the text processing. The words ”byggeansvar” and ”aktivitetsplan” will become ”byggeansv” and ”aktivitetspl”, and the words ”byggeansvarene” and ”aktivitetsplaner” will become ”byggeansvar” and ”aktivitetsplan”. This is because these words can not be defined as the same, and the amount of noise is unchanged in the text.

There are many libraries for finding the language used in a text. Nltk, Spacy, googletrans and polyglot are some of the NLP libraries that can detect language [25, 37, 10, 32]. All of them can detect whether or not the text is in English, Norwegian, or other languages. However, only polyglot can detect Norwegian Nynorsk, and other English regional variations can not be found. The method works best when the text to detect consists of a sentence or more. This is due to the fact that a lot of languages contain the same words. For example, Danish, Norwegian and Swedish are languages that are difficult to differentiate when looking up one or two words.

Another method for detecting language variations is to use word lists from the different languages and develop algorithms for detecting based on these. In this thesis, the word list approach has been used.

4.2.3 Distracting words

Distracting words are words that add little to no extra meaning to a text. They are words that give a low discrimination power between texts [16] and are often referred to as stopwords. However, stopwords are just one type of distracting words found in natural language texts — in this case exam answers — and can therefore be seen as a subset of the larger superset that is "distracting words". They often contribute to a more varied way of writing texts with identical meanings and are one of the natural language's most common noise sources. Therefore, they are often removed in NLP tasks to make it easier to compare texts. There are distracting word lists within most languages that are easily accessible on the internet. Removing these words from texts makes it easier to compare.

4.2.4 Synonyms

Synonyms are words with the same or a similar meaning to another word. In NLP, synonym words can cause variations between texts without a variation in the meaning. Examples of synonym words are "data" - "information", "nettside" - "nettsted", "hensikt" - "formål" and "programvare" - "software". If these occur a lot, it can be challenging to compare equally correct texts as equal.

Synonym words need to be detected to remove these text variations. Nltk wordnet with synset can be used for retrieving synonyms from the English wordnet [24]. PyDictionary is another synonym API for synonym retrieval [29]. However, it uses the website synonym.com to extract the synonyms, making it a bit slow. It also contains only English synonyms. There is no easily accessible synonym API for Norwegian. This thesis aims to create synonym detection with the inspiration from the PyDictionary did, but for Norwegian Bokmål text.

4.2.5 Spelling errors

Spelling errors can be defined as words spelled in a way not supported by the assigned written language. It can also be words that are defined as "correctly spelled" but is the wrong word. Spelling errors often occur within these permutations: insertion, deletion, substitution, and transposition of single characters [43]. Insertion is when an extra character is added to the word, for example, "Spellling" instead of "Spelling". Deletion is when a character is missing from the word, for example, "Splling" instead of "Spelling". Substitution is when a character is exchanged with another, for example, "Sbelling" instead of "Spelling". Transposition is when two characters swap places,

for example, ”Seplling” instead of ”Spelling”. The edit distance between two words, as mentioned in chapter 2.4, is a method often used for detecting spelling errors. C-rater, mentioned in chapter 3.2.1.1, uses the topic of the question to impact the spelling correction [18]. For example, if a question concerns US presidents and the word ”Reagons” is used, it would normally be corrected to ”Reasons” but is corrected to ”Reagan”.

One of the big problems is when a spelling error becomes a correct word. For example, ”Selling” instead of ”Spelling” will have an edit distance of 0 to ”Selling”, and no correction will happen. The method to correct these spelling errors is to use trained text models that detect the word that is most likely to occur around other words.

Textblob, Pyspellchecker, and Symspell are a selection of Python libraries used to detect and correct spelling errors. Textblob and Pyspellchecker utilize Peter Norvig's paper, ”How to Write a Spelling Corrector” [21, 30]. They cover multiple languages, but Norwegian is not one of them. Symspell, on the other hand, covers all languages. The only requirement is a list input with word frequencies for the given language [9].

4.2.6 Casing

In the written form, upper case letters are used at the start of a sentence, for proper names, and for official titles. Such as ”This is an English text written in Norway”. However, when the text is written in a spoken form, it can try to communicate a more profound meaning by using casing more creatively. For example, a word in only uppercase letters like ”STOP” indicates more importance, and the reader is more likely to focus more on the text around this word. On the other hand, a sentence like ”you are SO nice” can mean the opposite or have the intent of being sarcastic. Casing is used to imitate the intended pitch of the voice to make the meaning clear [47]. If this is important for the text meaning, casing should not be removed. However, all casing is reduced to lower case for most NLP tasks. For text such as school exams, this is also applicable.

4.2.7 Word lengthening

Word lengthening or word stretching is used in spoken form to visualize and amplify the intended meaning. For example, words like ”noooooo” and ”finallyyyyy” can strengthen the intended sentiment, often used in tweets or text messages. However, it is a spelling error if the intent is not to enhance the sentiment. Lengthening can be removed by first reducing all equal characters in a sequence only to contain two characters. Then use an edit distance calculation on the word. The use of lengthening to amplify the meaning in exam text is not likely to occur. However, it can happen unintentionally as a spelling error.

4.2.8 Word division and contractions

In Norwegian, it is common to contract more words than in English. It is a common mistake, especially when writing Norwegian with many English terms or vice versa. Should this example sentence be written as ”let har block scope”, ”let har block-scope”, or ”let har blockscope”? The sentence is in Norwegian but with the use of an English word. It leads to large quantities of word variations, and students disagree on which word variation is the correct one.

A method for removing this variation is to look up all bigrams and compare them to the unigrams in the text. If they both occur often, there is a chance they are used as the same word. Then the words can be corrected to one of the two variations.

4.2.9 Shortening of words

Shortening is the use of abbreviations and contractions of words and sentences. For example, the term ”function” can be written as ”func”, and the words ”let us” can be written as ”let’s”. It can be the use of acronyms such as ”Single Page Application” to ”SPA” or ”natural language processing” to ”NLP”. Some types of shortenings can be used in written form and some only in spoken form. A list with predefined shortenings will be the best option to remove this noise. A sensor of the exam can add acronyms specified for the system.

4.2.10 Distracting characters

Distracting characters can be symbols, punctuations, and numbers. Symbols are any special character used to express an idea, mood, or feeling [1]. Emojis are one type that can express a full meaning without using any words. Structured text such as mathematical formulas and code snippets are other types of symbol usage. Removing these types of noise are often a difficult task. The noise that emojis contribute to can be removed in two ways. Either remove the part containing the emoji or convert the emoji to a natural language word. The latter can only be helpful if the system requires these types of information from the text. For example, if the system analysis the semantics in texts. It is critical for mathematics and code parts to remove the whole element or nothing. This is due to added noise when cleaning away all forms of numbers and punctuations. If only some words are left in the structured text, it can not be used to analyze anything. For example, cleaning the code in figure 9 will give the sentence ”param let a return a param”.

```
param => {
  let a = 1;
  return a + param;
}
```

Figure 9: Illustration of distraction characters with code.

Removing punctuation and numbers in a typical natural language sentence is, on the other hand, a smart thing to do in most texts. For example, the words ”hi❤️” and ”hello!” can then easily be compared to the words ”hi” and ”hello”. In Python, removing punctuations and numbers can be done with regular expressions.

4.2.11 Grammatical errors

Grammatical errors are word combinations that does not fit the grammatical rules for the written language. Grammatical errors can occur as incorrect use of inflection in word form, use of punctuation, repetition of words, and more. To get a precise grammatical errors removal, a trained text model on a corpus is needed.

4.2.12 Word form and inflection

Word form is the use of different word inflection. It is also described as morphological variation in some systems [18]. This can contribute to noise in two ways. If the word is written grammatically incorrect in a sentence such as ”I program yesterday” instead of ”I programmed yesterday”. For Essay grading, grammar is often an essential element contributing to the given grade. Some systems will therefore need to detect this noise. The noise is useful because it can not be removed in the preprocessing pipeline due to the effect it needs to have in the grading model.

Word Form can also contribute to noise when different word inflections are interpreted as different words in information retrieval algorithms. The problem can be solved by changing all the words to their root form, also called normalization. For example, the word ”funksjonen” and ”funksjoner” both become ”funksjon”. In NLP, two techniques are used to normalize the word by removing inflection variations. Stemming and lemmatization are the two variants. Stemming works by removing the suffixes to reduce the word to its root stem. The method will make a lot of words that are not actual words or change the meaning. For example, the words ”universal”, ”university” and ”universe” are reduced to ”univers”, lemmatization will, on the other hand, ensure the word normalization to an actual word and, more often, the correct word. It uses a dictionary to reduce the word to a root form. For example, the words ”mest”, ”meste,” and ”mer” are reduced to ”mye”. Lemmatization is used in this study’s preprocessing pipeline.

4.2.13 Incorrect use of punctuation

The use of punctuation makes a text easier to read and communicates correct information. The most common punctuations in English texts are: full stops, commas, question marks, exclamation, colons, and semi-colons [47]. In NLP punctuation’s are often removed, however, it can sometimes remove the semantic or the sentiment of a text. For example, the text ”A woman, without her man,

is nothing" and "woman: without her, man is nothing" contains the same words, but the meaning is opposite. Since the analysis will be on Short Answer, where the content is most important, all punctuations will be removed in the final pipeline.

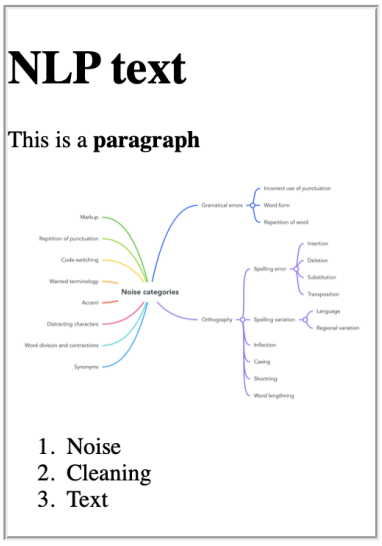
4.2.14 Repetition of words

Repetition of words or word sequences such as "variables **declared with declared with** let **have have** block-scope" can often happen unintentionally in an exam setting because of the fast writing. It will not add meaning to the text and can make the similarity algorithm weight the answer incorrectly. This can be removed by analyzing repetition in different n-grams. In the final pipeline, uni-gram repetition is removed.

4.2.15 Markup

Markup language such as HTML or LaTeX is used to combine both plain text and extra information in a text. It can add structure to a text with, for example, different font sizes or bullet points. A text can be bold, italic, or underlined, and images can be added with different tags. In NLP, only text can be processed, and the extra information in the answer must be removed. This causes some of the information to be lost. BeautifulSoup is a good tool for removing markup tags. Figure 10 illustrates how markup is removed.

```
<html>
    <body>
        <div style="border: groove">
            <h1>NLP text</h1>
            <p>This is a <strong>paragraph</strong></p>
            <img src="noisecategories.png" />
            <ol>
              <li>Noise</li>
              <li>Cleaning</li>
              <li>Text</li>
            </ol>
        </div>
    </body>
</html>
```



Text output: NLP text This is a paragraph Noise Cleaning Text

Figure 10: Illustration of how HTML markup looks.

4.2.16 Code-switching

Code-switching is when two or more languages are used in combination. This can be words from different languages in one sentence or within one word. ”Shall we go til huset” combines Norwegian and English words, while ”functioner” and ”applikations” use an English word with a Norwegian ending and a Norwegian word with an English ending. Sentence-based code-switching can detect the language used for each word. Just as mentioned in 4.2.2. Then convert some of the words so that all words are written in the same language. Code-switching within a word is more difficult to fix due to the unknown switching.

Language models can be used to detect code-switching [15]. The model needs to be trained on a corpus with the expected code-switching languages. In this theses, no method has been tested to remove code-switching.

4.2.17 Repetition of punctuation

Repetition of punctuation is the use of multiple punctuations to express a more profound meaning or amplify the meaning. For example, ”...” can indicate a waiting/pause or that something is stupid, and ”!!!!” can mean very important or a very strong exclamation. In Short Answers, this is rarely used or useful for the meaning. It is therefore removed just as other distracting characters, using regular expressions.

4.2.18 Wanted terminology

Wanted terminology is the use of key terms that will help a text score higher in the systems similarity algorithm. For example, if an answer contains a lot of the same terms as used in the reference answer, the student’s answer can often be interpreted as more correct. The problem with this is that it can contribute to a lot of false positive and false negative scores. This is because the correct words are used but in an order that makes the answer incorrect. Or the wrong words (according to the reference model) are used, but the correct meaning is written. To fix this noise type, the model has to use sentiment analysis instead of word-based analysis.

4.2.19 Accents

Accents are often used to stress one syllable over adjacent syllables [49]. Different types of accents are é, ó, ö, and ñ. These will often contribute to noise in NLP because the meaning is not that different without the accent. `unidecode.unicode("string")` in Python is a method for exchanging all Unicode strings with an accent to the closest possible representation in Ascii. This will remove all accents. Unfortunately ”å” is also an accent character. This is a needed character

in Norwegian and can not be removed. A self-defined dictionary with accent translations can also be used. This method will lead to more control in the text and is used in the final noise removal in the texts.

5 Dataset Analysis

It is essential to get familiar with the dataset to make a good preprocessing pipeline. What do the questions and answers look like, what types of noise exist in the answers, and how prominent are they? This chapter presents the structure of the dataset, the noisy elements, and the methods used to detect the noise. Based on chapter 4, some noise categories have been selected for deeper analysis.

5.1 Introduction to the dataset

In this study, two datasets are used. They are both retrieved from exams in the same course, "IT2810 Webutvikling" from the years 2018 and 2019. The course is a third-year subject primarily taken by students within the information technology and electrical engineering faculty at the Norwegian University of Science and Technology (NTNU). It is about web development, where they learn about frameworks, architectures, central languages, formats, and standards within web applications and web services [27]. The subject's language of instruction is Norwegian. The exams can therefore be written in either Bokmål or Nynorsk, but it is also allowed to write in English. Since the course is within the Computer Science field, many of the technical terms are English.

The questions vary quite a bit. Some of them ask about one concept, and others about multiple concepts. There is even a question in the dataset that asks the student to answer one of two concepts. Some require natural language examples (unstructured text) and others code examples (structured text). However, a common denominator is that all of the questions are open-ended.

The parameters that are used within this study of the dataset are;

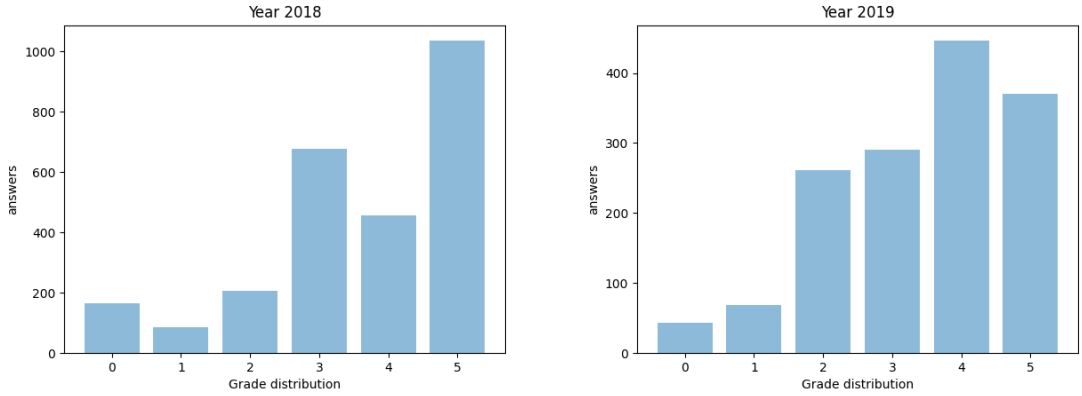
- Student answers that is formatted and stored as HTML.
- A score for each answer graded with a score from 0 to 5, where 5 is the best score, and 0 is the lowest score. 0 represents a grade of E and 5 a grade of A. The score distribution for exam 2018 is presented in figure 11a and exam 2019 in figure 11b.

The exam from 2018 consists of 16 questions, with 164 students answering each question. The exam from 2019 consists of 8 questions, with 185 students answering the questions. The average word length for exam 2018 is 74 words, and for exam 2019 is 122 words. Exam 2018 and 2019 have an average answer length within the range used to categorize Short Answers. Still, exam 2019 contains a lot of answers with a longer text average and is closer to being categorized as an Essay. Both exams, with all questions, are listed in the appendix A and B. As mentioned, the main difference between the exams is the answer length. The length difference is the major reason this study uses two datasets from the same course as a basis for the experiments. In the experimental

study, questions with a different answer length will be used. The average answer length for each question are presented in table 3 and 4.

	Number of question	Number of students	Average answer length
Exam 2018	16	164	74 words
Exam 2019	8	185	122 words

Table 2: Information about the dataset.



(a) Exam 2018

(b) Exam 2019

Figure 11: Grade distribution for the whole exams.

Q	Average word length	Average unique words (before preprocessing)
1	44	29
2	36	28
3	84	57
4	188	74
5	48	37
6	49	37
7	98	70
8	22	18
9	74	49
10	51	38
11	58	43
12	70	47
13	141	87
14	58	43
15	102	67
16	73	52

Table 3: Exam 2018: Word length statistics.

Q	Average word length	Average unique words (before preprocessing)
1	98	68
2	121	75
3	119	75
4	183	107
5	72	42
6	56	38
7	223	127
8	103	69

Table 4: Exam 2019: Word length statistics.

5.2 Visualizing the data

A dashboard was created to get an overview of the answers in the dataset. It contains two pages. The first is a visualization of the original answer, the automatically processed answer, and a list of the changes done in the preprocessing presented in figure 12. The second page visualizes statistical features in the answers presented in figure 13. It is possible to navigate through all of the questions and look at the question asked on both pages.

Originally, the dataset was given in a JSON format, making it challenging to get familiar with the dataset. By implementing a visualization page, this problem was solved. Now it was possible to get an overview of the dataset. This overview made it easier to understand the existing noise sources in the dataset and find good answers to use in the experiments.

The second page visualizes the statistics from each of the questions. The statistical features that were the most useful were score distribution and the most used n-grams. These features were used to choose the answers for the experiment. The n-grams were used to look for frequently used terms. If there were a distinct pattern, it would be easier to use due to the similar way of writing. The n-grams were also used to ensure the reference answers used the same types of words as the answers did.

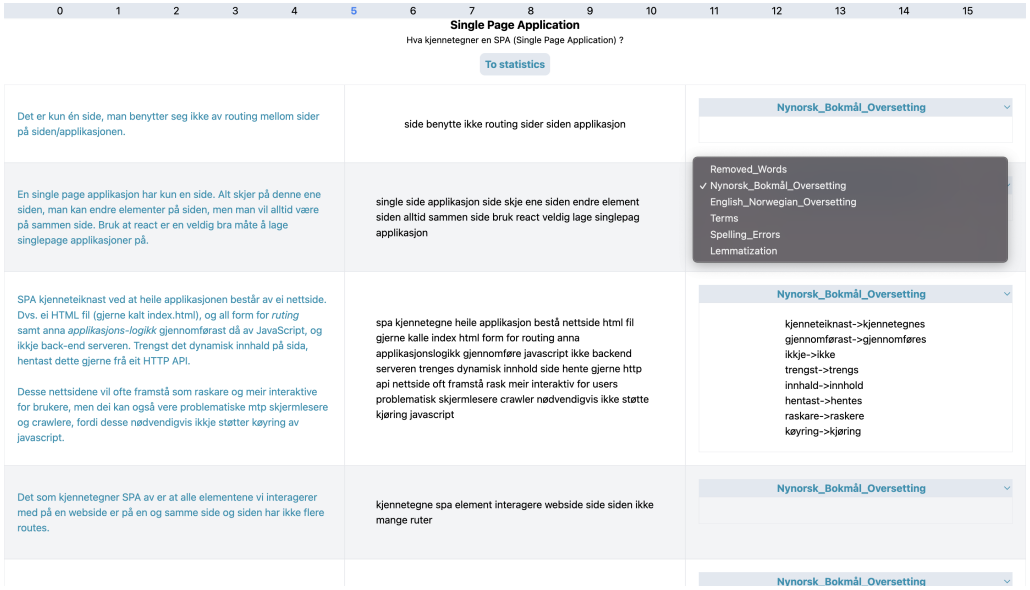


Figure 12: First page of the dashboard.

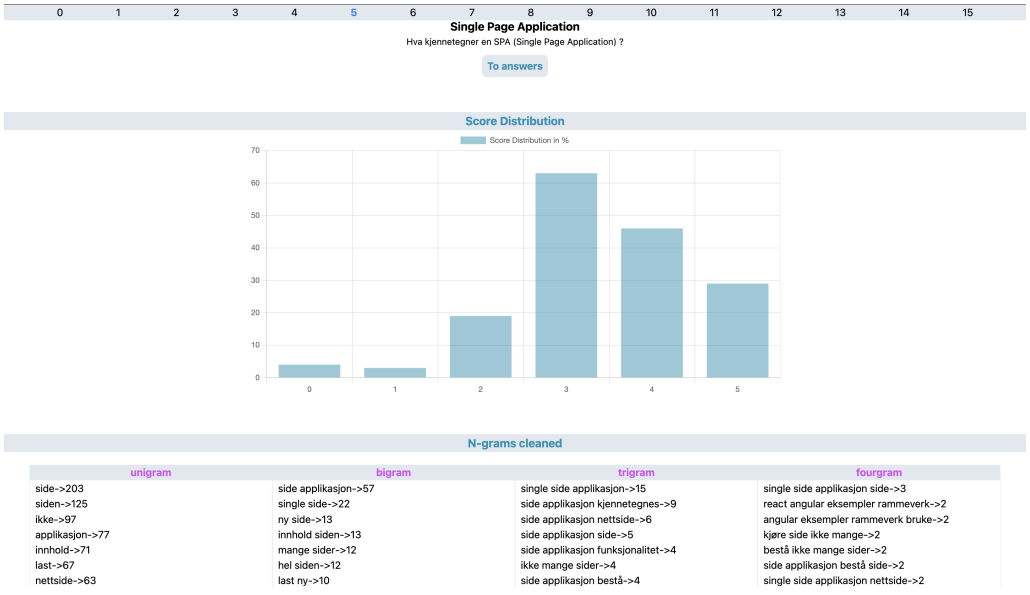


Figure 13: Second page of the dashboard.

5.3 Noise analysis of the dataset

Noise removal — also called text cleaning, text normalization, or text preprocessing — is an essential task when analyzing any form of natural language [4, 18]. It is often a task that is less appreciated in the development of automatic grading systems. Lots of the research papers within this area choose not to mention this step in thorough detail and focus more on other steps within the grading pipeline [22, 53, 33, 43, 12]. A great deal of text preprocessing libraries have been developed through the years. Nltk, Gensim and Spacy are some of them [25, 31, 37]. They support multiple languages, but work best with English texts. Many of the methods do not exist within the library or work less precisely when working with Norwegian text. There is little research on Norwegian NLP, and the tools are limited. It is also uniquely special because the combination of regional variations in Norwegian text is high.

A thorough noise analysis is conducted to understand better the noise sources and which of them that is the most prominent in the answers. It is based on the noise described in chapter 4.2. However, not all of them are analyzed further due to the low probability of occurrence in a typical Computer Science exam. The noise categories in this chapter and chapter 4 are not all presented with the same name, as this analysis is more specified towards this dataset. Table 5 shows the names with the corresponding name in the chapters.

5. Analysis dataset	**4. Noise categories**
Stopwords	Distracting words
Language and regional	Spelling variation
Spelling errors	Spelling errors
Normalization - lemma	Word Form and Inflection
Punctuation and numbers	Distracting characters
Question words	Wanted terminology
Synonyms	Synonyms

Table 5: Chapter names with corresponding noise category

5.3.1 Stopwords

A text consist of around 35-45% stopwords [16]. By removing stopwords from the text, a large part of the text is gone, and the remaining text consists of words with higher discrimination power. Identifying the percentage of meaningless words these exam texts contain will give a better impression of how impactful this noise source may be.

Stopword lists used in natural language processing are normally retrieved from a ”general predefined” stopword list. However, almost every NLP task needs to modify its stopword list to better match the task in mind [16]. Regarding the dataset in this study, the final stopword list

contained both Norwegian Nynorsk, Norwegian Bokmål, and English stopwords. The stopwords were retrieved from these two stopword lists [8, 3], and modified to fit the dataset better.

By analyzing the most frequently used words, some words were removed from the stopword list because of the meaning they contributed to in the text, and others were added to the stopword list because they added noise to the text. Since the answers in the exam have contextual importance, the negations cannot be in the stopword list. All words meaning ”not” is removed. This includes ”ikke”, ”ikkje”, ”not”, ”no”, and ”nor”. Removing these words from the text can cause an opposite semantic meaning of the text. Other words that are removed from the stopword list are ”this”, ”siden”, ”var”, as they are words that can be important when answering some of the questions. Words have been added that have a different inflection of words that are already in the text. Some abbreviations have also been added to the list. These are words like ”eks”, ”osv”, ”etc”, ”via” and ”mtp”.

The list contains a total of 345 words. Where 226 are Norwegian words and 119 are English words. Both of the exams have approximately the same average percentage of stopwords. An answer from exam 2018 contains an average of 43% stopwords, and exam 2019 has an average of 46%. Words with low to no comparison capabilities contribute to almost half of the exam answers. Table 6 presents the average number of stopwords, the lowest percentage of stopwords in an exam (without counting the not answered answers), and the maximum percentage of stopwords in an exam answer. The low minimum stopword percentage can be due to a concise answer, a very short answer, or an answer listed as a bullet list. The high maximum stopword percentage can be due to an unnecessary long text where much of the answer is unnecessary when answering the question.

	Average %	Min %	Max %
2018	43%	4%	75%
2019	46%	6%	66%

Table 6: Average, minimum and maximum percentage of stopwords in an answer, for each exam.

5.3.2 Language and regional variations

To find out the degree of noise in languages and regional variations, a count of all answers containing Nynorsk, Bokmål and English was done. Three dictionaries, one for each language, were used to determine if an answer contained words within the language. When counting Nynorsk and Bokmål, a word was only counted as Nynorsk when the word did not exist in Bokmål dictionary as well. The Nynorsk and Bokmål dictionaries were retrieved from Nationalbiblioteket, the versions were updated in 2019 [39, 38]. The English dictionary was retrieved from a corpus developed on the most used words in 12 different English word lists, the version was updated in 2018 [48]. The Nynorsk dictionary contained 392 920 words, the Bokmål contained 583 876 words, and the

English contained 181 000 words. Although the English dictionary is small, it includes the more likely words to occur. However, many English technical terms are not in the dictionary and are not counted.

	Total answers	Not answered	Total Bokmål	Total Nynorsk	Total English
2018	2 624	22	2 599 (99.0%)	66 (2.5%)	1 177 (44.9%)
2019	1 480	16	1 463 (98.9%)	9 (0.6%)	574 (37.8%)

Table 7: Language and regional variants in the texts. All answers that contains at least one word from the language in each column is counted as containing the language.

Table 7 contains the results. 99.0% and 98.9% of all answers contain words defined as Bokmål. 44.9% and 37.8% of all answers contain words defined as English. 2.5% and 0.6% of all answers contain words defined as Nynorsk. This indicates the importance of looking at language variations in the system. Looking at Nynorsk might not have a huge impact on a system's precision. However, it is an important step for Norwegian exams due to the unknown amount of Nynorsk answers for each exam. It may be even more important for universities on the West side of Norway to handle this noise due to the larger degree of the population writing Nynorsk.

5.3.3 Spelling errors

In this noise analysis, errors are words that can not be categorized within one of the three language dictionaries or a word that ten or fewer students use. This means that not all words that are defined as errors are actually errors, but they are words that are not as common. It can be due to the spelling form or that it is a technical term. In this analysis a word is considered as correct if used by 10 or more students.

In exam 2018, 70% of all students' answers have written at least one word defined as an error, and exam 2019 has 84%. The average percent of errors in an answer is 3.8% for exam 2018 and 3.3% for exam 2019.

	Total answers	Total answers with errors	Average percent of errors in an answer
2018	2 624	1 859 (70%)	3.8%
2019	1 480	1 249 (84%.)	3.3%

Table 8: Error data from the answers.

5.3.4 Normalization - lemma

Writing an entire text in root form would not make sense, but using different inflections would make it harder for a computer to treat words as equal. Therefore, reducing the words to their root form can remove noise in the text. For this analysis, the normalization technique that is used is lemmatization. A count has been done to determine how many of the words in the text that are not written in root form. The library Spacy [37], with the Norwegian corpus ”nb_core_news_lg”, has been used to develop a lemmatization dictionary with all words not written in root form in the exams. Spacy supports three corpus sizes. The analysis uses the largest corpus to get the most precise lemma predictions. In exam 2018, 26 683 words are converted to root form. There are 1 869 unique inflections and 1 252 unique root words from these words. This means that 617 unique words can be counted as their other inflections. In exam 2019, 24 344 words are converted to root form. From these words, there are 1 669 unique inflections and 1 167 unique root words. This means that 502 unique words can be counted as its other inflections. For example the words ”variabelen”, ”variablene”, and ”variabler” was converted to ”variabel”.

	Total words that is lemmatized	Total unique words that is lemmatized	Unique root-words	Average percent of lemma in an answer
2018	26 683	1 869	1 252	14.4%
2019	24 344	1 669	1 167	10.5%

Table 9: Inflection data in the answers.

5.3.5 Punctuation and numbers

In natural language, punctuation and numbers are mainly used to stop a sentence with the use of ”.” ”,” ”!” and ”?”. However, many of the questions require the students to come up with a code example. Code is a structured text type, and analyzing it with the same method as for natural text will give poor results. The table 10 presents an average of punctuation used per exam, average numbers used, and average typical code punctuation. These are all punctuations except those mentioned at the start of this paragraph. All punctuations are counted as individual characters. This is also the case for multi-digit numbers. The measures can tell something about the amount of structured vs. unstructured text in the answers and tell why a similarity algorithm performs poorer in some cases. The low number of code punctuations in relation to the total number of punctuation can indicate that no code has been used and vice versa.

	2018	2019
Average use of symbols	18.4	25.0
Highest symbol average for a question	46.5	35.6
Lowest symbol average for a question	6.9	13.6
Average use of numbers	0.72	1.4
Highest number average for question	2.15	4.1
Lowest number average for question	0.04	0.14
Average use of code symbols	8.8	9.4
Highest code symbols average for question	27.4	26.4
Lowest code symbols average for question	1.5	3.6

Table 10: Symbol use in the answers.

5.3.6 Question words

Question words are words presented in the question. It is a form of wanted terminology and can easily contribute to false positives and false negatives. Table 11 and 12 present the average percentage of question words in an answer on a per question basis. All stopwords (according to the definition in 5.3.1) are removed in this measure. The questions have significant variations of question words used in the answers. Exam 2018 varies from 3.8% to 55.2% with an average between the questions of 17.7%. Exam 2019 varies from 11.6% to 17.6%, with an average between the questions of 16.9%. The number of question words seems to be the same regardless of the text length. This can indicate that the question words are needed to answer the question. Removing them may disrupt the contextual meaning, and using them as scoring words may be beneficial for some and disrupting for others. The use of synonym words may minimize this problem.

Question	% of answer that is question words
1	36.7%
2	27.1%
3	11.2%
4	19.8%
5	9.1%
6	9.0%
7	13.1%
8	55.2%
9	16.1%
10	11.8%
11	13.6%
12	25.1%
13	13.3%
14	3.8%
15	6.9%
16	11.6%

Table 11: Exam 2018: Average amount of words used that are from the question.

Question	% of answer that is question words
1	14.7 %
2	21.6%
3	15.7%
4	14.7%
5	26.9%
6	12.4%
7	11.6%
8	17.6%

Table 12: Exam 2019: Average amount of words used that are from the question.

5.3.7 Synonyms

Extracting synonyms proved to be a time-consuming task. No API for Norwegian synonyms was easily accessible. Nor any lists with synonyms. The only method I could find for gathering synonyms was to use the website "Synonymordboka.no". With the inspiration from the PyDictionary mentioned in chapter 4.2.4. The words that I wanted to find synonyms for were searched for with this website. For exam 2018, a total of 4757 unique words were sent through to find synonyms and 4862 unique words for exam 2019. For each word, a list of possible synonym words was returned. From the list, only the synonym words that also existed in the answers were gathered. Based on these, a "synonym - dictionary" was created. The 2018 exam dictionary contains 2042 words, connected to a list with possible synonym words, and the 2019 exam 2158 words. The list of possible synonyms for each word often contained more than 10 words. Some of the words were good synonym words, and some were not. The words "nettside" - "nettsted", "designer" - "utvikler", and "data" - "datamaskin" or "data" - "info" are useful synonyms. The words "utforming" - "smart" and "transport" - "kommunikasjon" is harmful and should not have been categorized as synonyms.

$$data = \{beskjed, datamaskin, forklaring, info, kjennskap, pc, server, tips, tjener\}$$

$$designer = \{skaper, utformer, utvikler\}$$

$$utforming = \{dyp, fasong, figur, form, oppbygning, preg, skape, skjult, smart, sort, versjon\}$$

$$nettside = \{nettsed\}$$

$$kildekode = \{applikasjon, kode, program\}$$

$$transport = \{bevegelse, fjerning, flytting, kjring, kommunikasjon, last, overfring\}$$

Figure 14: Examples of extracted synonyms.

One of the big problems is how the system shall find useful synonyms. The lists of possible synonyms are enormous, and taking a random word from the synonym list is not safe. A manual step where a person goes through the synonyms and picks good synonym combinations may work. This will, however, take a long time and defeat the purpose of automatic grading. Another problem is that even if the word is a good synonym in one setting, it may not be a good synonym in another setting. For example, in some settings, "data" means "datamaskin" in others, it means "info", so choosing the correct one is essential.

There is a considerable risk of using the synonyms that were extracted in this research in the processing pipeline, and it will, therefore, not use synonyms.

5.3.8 Measuring total noise

To get an overview of the total amount of noise present in the dataset, two calculations of noise were made. In this context, noise is defined as any word that was modified by the pipeline. Both measures an average percentage of noise in the answers, however different parameters of noise were used. The first method looks at all noise present in the answers as a percentage of all words. This includes the following parameters: English written words, Nynorsk written words, words defined as errors, words not written in root form, and stopwords. First, a percentage was calculated by summing up all words classified as noise divided by all words in the answer. Then the percentage was grouped by grade to see if a noise pattern existed. This measure is referred to as "Average Noise".

The second method looks at noise present in the answers as a percentage of words without the stopwords. The parameters are: English written words, Nynorsk written words, errors, and words not written in root form. All noise types as in the first measure except stopwords. Stopwords are easily defined and removed, and an analysis of the other noise types is more interesting. In this percentage, calculation words are summed up and divided by total words in the answer minus stopwords. This percentage is then also grouped by the grade. This measure is referred to as "Average Complex Noise".

Table 13 presents the percentage of noise in an answer as an average of all answers from all questions. The Average from question to question is much higher, and therefore presenting it as "Average Noise" shows a higher percentage of the text for exam 2018 with 60.44% and 2019 with 58%. The "Average Complex Noise" exam in 2018 was 34.4%, and the exam in 2019 was 29.5%. This shows that the non-stopword related noise is quite extensive. A text contains around 30% of noise after all stopwords are removed.

Exam	Average Noise	Average Complex Noise
2018	60.44 %	34.40%
2019	58.00%	29.59%

Table 13: Average amount of noise in an answer for each exam.

Table 14 and 15 presents the noise divided by score from the two exams. The average noise for the scores between 1 and 5 is almost the same for both exams. However, the score of 0 is significantly lower. An explanation can be that a poor answer is written in a manner where neither correct information nor noise is present.

Score	Average Noise	Average Complex Noise
0	42.73%	24.24%
1	62.18%	35.09%
2	64.54%	36.67%
3	64.29%	36.07%
4	64.22%	37.19%
5	64.69%	37.11%

Table 14: Exam 2018: Average amount of noise divided by grade. Average noise is all noise sources in the text. Average complex noise is the noise that are not easily fixed.

Score	Average Noise in answer	Average Complex Noise
0	30.61%	15.57%
1	61.01%	29.14%
2	63.00%	31.63%
3	64.98%	33.09%
4	64.02%	34.32%
5	64.35%	33.77%

Table 15: Exam 2019: Average amount of noise divided by grade. Average noise is all noise sources in the text. Average complex noise is the noise that are not easily fixed.

6 Experiments

These experiments aim to establish if preprocessing affects the final comparison with a reference answer. If this is shown to be true, which of the categories has the most significant impact?

The literature shows that a direct comparison between student answers and reference answers will not always provide a good correlation and will therefore lead to an incorrect grade [18, 4]. Therefore, a very good answer can be discarded and receive a lower grade, as the language or presentation is very different from the reference answer. On the other hand, a higher correlation, and therefore a higher grade than the answer deserves, can also occur if an answer uses the same language and presentation as the reference answer. Little research has been done on the Norwegian language, and few libraries support preprocessing of the Norwegian language or a combination of Norwegian and English language. Therefore, before advances in research on algorithms that compare answers are possible, it is crucial to establish good preprocessing knowledge and algorithms.

Since the focus is on preprocessing, a fairly simple comparison algorithm is used and does not require a corpus in a specific language or a trained language model. The absolute values of the comparison algorithm are not important, but the trend toward "more or less" like the reference answer is sufficient to conduct and conclude on the experiments.

The goal of the experiments can be split into two parts as follows:

1. Establish if the combined set of noise-cleaning leads to a more accurate comparison with the reference answer.
2. Establish which of the steps in the noise-cleaning algorithm that is most impactful, measured by the amount of noise removed and the effect on the final correlation.

6.1 Preprocessing pipeline

Based on the noise analysis in chapter 5.3, a text preprocessing pipeline has been created. This pipeline contains both big and small steps to remove noise from the answers.

Figure 15: Pipeline for preprocessing.

The elements in the preprocessing pipeline, as illustrated in figure 15 are described in detail in the following sub-chapters.

6.1.1 HTML tag removal

The tool BeautifulSoup [36] is used with the feature ”html.parser” to remove all HTML. Just the text within text tags is returned. All figures, illustrations, and special tags such as bullet points and bold text are therefore removed.

6.1.2 Character level preprocessing

This step contains a lot of small but easy preprocessing. First, all characters are reduced to lower case. Then, all punctuation and numbers are removed. There are some special rules for some of

the characters. For the character ”-”, the word before and after it is combined. For the character ”/”, the word before and after is divided. Any redundant whitespace is also removed. All of these operations are done with the use of regex substitution.

6.1.3 Strip accent

Words such as ”én” or ”värt” are reduced to ”en” and ”vært”, respectively. These operations are done using regex substitution and a list of accent translation characters.

6.1.4 Stopword removal

All words in the stopword list mentioned in 5.3.1 are removed. Since the average number of stopwords in the answers is between 43%-46%, it is essential to remove them early in the pipeline to avoid unnecessarily processing large amounts of text.

6.1.5 Nynorsk to Bokmål translation

The library Apertium [2] is used to translate Nynorsk to Bokmål. Apertium is an open-source translation library with support for Nynorsk, Bokmål, and many other languages. A Nynorsk-Bokmål dictionary, with all unique words only present in Nynorsk, was developed with the use of all words in the dataset.

6.1.6 English to Bokmål translation

The Python library googletrans [10] (a wrapper around Google Translate) is used to translate English to Bokmål. An English-Bokmål dictionary was developed with all the English words in the dataset. However, not all terms defined as English were translated. Only words that were used in both Norwegian and English were translated. These are the words that contribute to more noise. A comparison between the equivalent word was done to evaluate which of the words required translation. First, the translation needed to be used by more than 5 students. Second, if the Norwegian word was used the most, a translation from English to Norwegian was done. The opposite was done if the English word was used the most. In this way, according to most students, the correct word was used in the translation.

6.1.7 Term identification

A lot of the words in the answers are technical terms that are not defined in any of the word lists used in the analysis. This is due to the difficulty of finding lists containing a lot of these terms. If the words are not defined as correct, they will go through the spelling corrector and become

another word. It is therefore important to find words that might be technical terms. All words that are defined as errors at this point in the pipeline are counted on a per-student basis. If more than 10 students used a word, it is counted as a technical term. These terms are then compared to find different spelling variations within the terms to reduce the number of terms. The words not defined as terms are sent through the spelling corrector.

6.1.8 Error correction

Spelling corrections can quickly become complex and are slow to generate. With the use of all 4 types of spelling errors: insertion, deletion, substitution, and transposition, enormous amounts of possible spelling errors exist. According to the SymSpell documentation, *"An average 5 letter word has about 3 million possible spelling errors within a maximum edit distance of 3, but SymSpell needs to generate only 25 deletes to cover them all"* [9]. The preprocessing uses SymSpell to correct the words. It is a spelling correction algorithm where only deletes are required. Transposes, substitutes, and inserts of the input terms are transformed into deletes of the dictionary term. SymSpell needs a dictionary with word frequencies to generate the correct words [7]. An edit distance of 2 is used to find valid words. For example, the word "størreslen" is corrected to "størrelsen" and the word "boostrap" to "bootstrap". However, this method is not flawless. The spelling correction will guess the most likely word, but it may guess incorrectly. Since there is no manual check whether or not the closest word is the correct word, a lot of them are changed to an incorrect word. Words such as "tekstboks" will be corrected to "tekstbok", "scalere" to "smalere" and "htmlen" to "himlen". The words that are often incorrectly changed are technical terms because the algorithm does not recognize these. If this happens, it will create a lot of new noise making. Therefore, the identified terms are added to the word frequency list with a high frequency to avoid making more noise. In experiment 2 the correctness of the error correction is tested, but for experiment 1 it is used as described here, hoping that it removes more noise than it adds.

6.1.9 Stopword removal

Another stopword removal is done to ensure all stopwords are removed. It is done a second time because the other processes may have corrected words so that they now reveal new stopwords, written in a form not in the stopword list.

6.1.10 Normalization

The words are reduced to their root form using the normalization dictionary as mentioned in chapter 5.3.4.

6.2 Experiment 1

This experiment aims to find out if text preprocessing and noise removal contribute to a better comparison of Norwegian computer science exam answers. A total of 10 answers from 2 questions in exam 2018 and 2 questions in exam 2019 were selected. This results in a total of 40 answers used in the comparison. These answers were compared with a reference answer using a cosine similarity algorithm with a term frequency weighted model. For comparing the effect of noise removal, different versions of the exam answers were compared with the reference answer: The raw answer with no preprocessing, a manually preprocessed version, and an automatically preprocessed version using the developed pipeline specified in section 6.1. These results were then compared to the manually processed answer (as a baseline) to figure out the effect of the preprocessing.

Somewhere between these results, we can expect to find the results of the developed algorithm, and the goal for the algorithm is to come as close to the manually edited answer as possible. Therefore, in addition to measure the correlation with the reference answer, we also measure the percentage of noise removed and the percentage of noise the automatic algorithm cannot remove (compared to the manual noise removal).

Hypothesis: Manual noise removal by an educated human will be close to perfect. Therefore, it should perform better than the automatically processed answers, which should achieve a higher correlation with a reference answer than the raw, unedited answer (referred to as original).

6.2.1 Method

This experiment requires a selected set of answers from the dataset. Due to the time-consuming task of manual text processing, a subset was used to reduce processing time. Also, not all questions and answers were usable for comparison. The method section presents a set of criteria for selecting questions and answers, a guideline for manual preprocessing, examples of manual and automatic preprocessing, the method used for the reference answer, and the scoring algorithm.

6.2.1.1 Rules to select questions and answers

The questions and answers used in this experiment were selected based on several criteria:

1. Questions requiring examples in the form of code or examples that are likely to vary significantly between answers were not used.
2. The question should be worded such that there is a reference answer. Questions asking for an opinion or a response to multiple concepts were not used.
3. The answers should contain a noticeable amount of noise to get the best possible comparison between the different processings.

4. The answers received one of the two highest scores (either 4 or 5), so the comparison should achieve a high correlation with the reference answer.

5. Different average length of the answers. All of the answers are within the length of a Short Answer. However, some of the answers are on the higher end of Short Answers. A more in-depth analysis can be done by selecting answers from both the higher and the lower spectrum.

The questions used were number 1, ”Variables declared with var and let”, and number 6, ”Single Page Application”, from exam 2018. They had an average answer length of 44 and 49 words, respectively, and a clear and defined concept to answer. Exam 2019 questions number 1, ”Responsive Webdesign”, and number 8, ”Global state management”, were used. They were double the average answer length with 98 and 103 words, respectively.

6.2.1.2 Rules for manual preprocessing

In manual preprocessing, all noise a person can find is removed. The following list defines some rules for the manual processing:

- Remove stopwords.
- Remove numbers, symbols, and punctuation.
- Set text to lower case.
- Fix spelling errors:
  - Find the correct word for all words written with insertion, deletion, substitution, and transposition of characters.
  - Remove any code-switching.
  - For each word: choose English or Norwegian written form. This applies to all answers.
- Normalize words:
  - Singular noun.
  - Infinitive verb.
- Define, for each word, the hyphenation that applies to all answers. For instance, the words ”block scope” and ”blockscope” are used a lot. In the manual processing, they are set to ”blockscope”.
- All synonym words are reduced to the same word. ”nettside”, ”nettsted” and ”webside” are all changed to ”nettside”.

6.2.1.3 Examples of preprocessing (original, automatic, and manual)

Variables declared using var and let - question 1, exam 2018

Original question (Norwegian Bokmål):
Forklar kort hvilket scope som gjelder for variabler som er deklarert med nøkkelordet **var** og hvilket scope gjelder for variabler deklarert med nøkkelordet **let**?

Original	
Variabler deklarert med let har block-scope Variabler deklarert med var har function-scope. Block-scope er ofte mer snevert enn function-scope da funskjoner kan ha flere blokker. Dette gjør at sannsynligheten for uønsket endring av variabler minker	
Automatic	**Manual**
variabel deklarere let blockscope variabel deklarere var functionscope blockscope ofte mye snever functionscope funskjone mange blokk sannsynlighet for uønsket endring variabel minke	variabel deklarere let blockscope variabel deklarere var funksjonscope blockscope ofte mye snever funksjonscope funksjon mange block sannsynlighet uønske endring variabel minke

Table 16: Question 1, exam 2018: An original answer used in the experiment, with the automatic and manual text processing.

Single Page Application - question 6, exam 2018

Original question (Norwegian Bokmål):

Hva kjennetegner en SPA (Single Page Application)?

Original	
SPA kjenneteiknast ved at heile applikasjonen består av ei nettside. Dvs. ei HTML fil (gjerne kalt index.html), og all form for ruting samt anna applikasjons-logikk gjennomførast då av JavaScript, og ikkje back-end serveren. Trengst det dynamisk innhald på sida, hentast dette gjerne frå eit HTTP API. Desse nettsidene vil ofte framstå som raskare og meir interaktive for brukere, men dei kan også vere problematiske mtp skjermlesere og crawlere, fordi desse nødvendigvis ikkje støtter køyring av javascript	
Automatic	**Manual**
spa kjennetegne heil applikasjon bestå nettside html fil gjerne kalle index html form for routing anna applikasjonslogikk gjennomføre javascript ikke backend serveren trenges dynamisk innhold side hente gjerne http api nettside oft framstå rask meir interaktive for users problematisk skjermlesere crawle nødvendigvis ikke støtte kjøring javascript	spa kjennetegn hel applikasjon bestå nettside html fil kalt indexhtml form ruting applikasjonslogikk gjennomføre javascript ikke backend server dynamisk innhold nettside hente http api nettside ofte fremstå raskere mer interaktiv bruker problematisk skjermleser crawler nødvendigvis ikke støtte kjøre javascrip

Table 17: Question 6, exam 2018: An original answer used in the experiment, with the automatic and manual text processing.

Responsive web-design - question 1, exam 2019

Original question (Norwegian Bokmål):

1. Hva er responsiv web design?

2. Nevn forskjellige teknikker som brukes for å implementere responsiv webdesign.

Original	
1. Responsivt web-design handler om å designe nettsiden sin slik at den responderer/skalerer bra på ulike vindus- og skjermstørrelser. 2. Ulike teknikker for å implementere responsivt webdesign er eksempelvis bruk av Viewport og eller Media-queries. Media-queries går ut på å sette ulike breakpoints for å definere ulik oppførsel/design for nettsidens i forskjellige størrelser. Kan si at for eksempel når nettisden blir mindre enn 500 piksler i bredde vil vi at dette skal skje. Bruk av CSS Flexbox eller Grid kan også brukes til å oppnå resposnivt webdesign. Her kan du for ekesmpel anngi bredde og høyde i prosent eller sette den til auto, for at det skal skalere i forhold til siden	
Automatic	**Manual**
responsiv webside handle designe nettside respondere skalere ulike vindu skjermstørrelse ulike teknikk for implementere responsiv webside bruk viewport mediaqueries sette ulike breakpoints for definere ulik oppførsel design for nettside forskjellig størrelse for nettside små piksel bredde skje bruk css flexbox grid bruke oppnå responsiv webside for angi bredde høyde prosent sette auto for skalere forhold siden	responsiv webdesign handle design nettside respondere skalere ulik vindsstørrelse skjermstørrelse ulik teknikk implementere responsiv web design viewport media queries sette ulik breakpoint definere ulik oppførsel design nettside forskjellig størrelse nettside mindre bredde bruk css flexbox grid bruke oppnå responsiv web design angi bredde høyde prosent sette auto skalere forhold side

Table 18: Question 1, exam 2019: An original answer used in the experiment, with the automatic and manual text processing.

Global state management - question 8, exam 2019

Original question (Norwegian Bokmål):

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

Original	
React kan man ettersom størrelsen på applikajsonen vokser få større og større problemer med håndering av state uten bruk av global state management. Man vil ofte ende opp med enten mange komponenter som har og håndterer state eller en/få komponent(er) som hånderer state. I det første tilfellet vil det være vanskelig å holde oversikt over hva staten er og man vil kunne få store problemer med flyt av data opp og ned i komponenttreet. I det andre tilfellet vil man ha stor og ukontrollerbar flyt av data mellom komponenter i treet ettersom alt må inn til hoved kompnenten(e) til slutt. Global state management løser disse problemene med at man har en global state ofte kalt store (redux) der komponenter kan hente ut og dispatche endringer etter behov uten å for eksempel måtte sende med props mange ledd ned i komponenttreet. Man har også mye lettere oversikt over hva staten er ettersom man (ofte) har en global state og alle komponenter forholder seg til samme state	
Automatic	**Manual**
react ettersom størrelse applikasjon vokse stor problem håndtering state uten bruk global state management oft ende enten komponenter håndtere state komponent håndtere state første tilfelle vanskelig holde oversikt stat stor problem flyt datum komponenttre andre tilfelle stor ukontrollerbar flyt datum komponenter tre ettersom hoved komponent slutt global state management løse problemene global state oft kalle stor redux komponenter hente dispatch endring behov uten for sende props ledd komponenttre mye lett oversikt stat ettersom oft global state komponenter forholde state	react ettersom størrelse applikasjon vokse større problem håndtere state uten bruk global state management mange komponent håntere state komponent håndtere state først vanskelig holde oversikt state mye problem flyt data komponenttre andre tilfellet mye ukontrollerbar flyt data mellom komponent tre hoved komponent slutt global state management løse problem global state kalt store redux komponent hente dispatche endring behov uten sende props mange ledd komponenttre mye oversikt state ettersom global state komponent forholde state

Table 19: Question 8, exam 2019: An original answer used in the experiment, with the automatic and manual text processing.

6.2.2 Reference answer

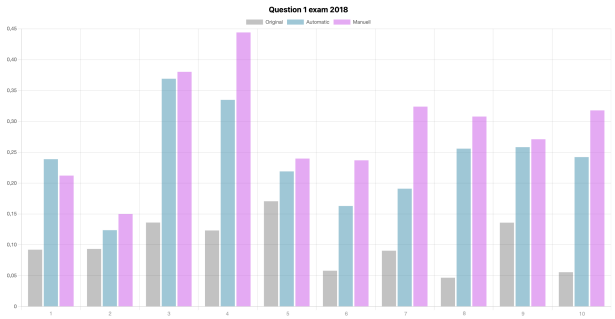
The reference answers used in the experiment were extracted from the exam's model answers. They were then altered to fit the 10 student answers used in the experiment. Typically, good reference answers are an essential part of a precise automatic grading system. However, in this experiment, the goal is not to get the most accurate grading (similarity comparison) but to analyze the effect of noise reduction on the answers. Furthermore, due to the lack of a good synonym word list (especially regarding technical terms), the reference answers must be adjusted to better match the vocabulary in the students' answers. Therefore, an n-gram analysis of the students' answers was conducted to identify the students' vocabulary. These results were then used to correct the reference answers manually.

6.2.3 Scoring algorithm

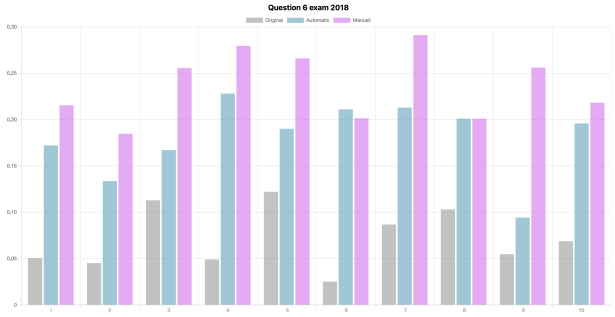
The scoring algorithm used to measure the student answer's correctness with regards to the reference answer is a cosine similarity algorithm (2.3). Vectors are generated with the Term Frequency of uni-grams and bi-grams in the texts. Term Frequency (without the IDF part) is used because the scoring algorithm looks for the usage of specific words, and the fact that they show up often should not negatively impact the score, but rather the opposite. TF-IDF would have weighted the terms lower because they show up often. Uni-grams were used to find unique words, and bi-grams were used to measure contextual meaning in the texts. Correct use of bi-grams was therefore rewarded and scored higher. This was done to counteract the correct use of words in the wrong order. Tri-grams were not used due to the low similarity found between student answers and reference answers. This method for scoring the correctness of the texts can not be validated as a suitable grading method due to its low similarity matching and higher possibilities for false positives and negatives. Still, it can be a qualified method to measure the improvement the preprocessing pipeline had on the texts.

6.2.4 Result

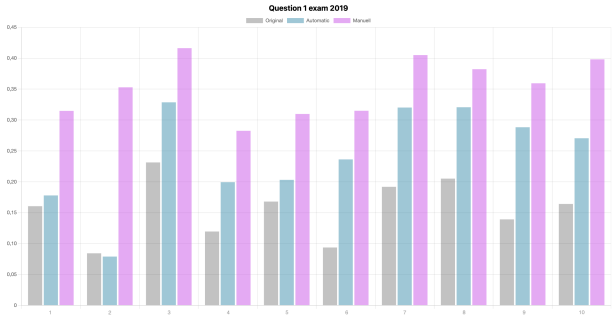
Figure 16a, 16b, 16c, and 16d presents the similarity score to the reference answer for the 3 processing methods for each of the 4 questions. The figures are listed in appendix C in a higher resolution if the reader wants to look at them in more detail. Almost all of the manually preprocessed answers performed the best, while the automatic answers performed, on average, somewhere in the middle, and the raw answers performed worst.



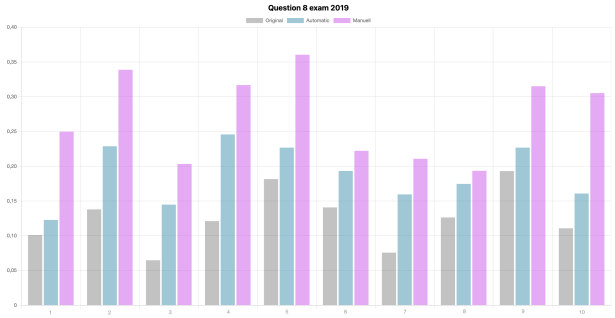
(a) Question 1 exam 2018



(b) Question 6 exam 2018



(c) Question 1 exam 2019



(d) Question 8 exam 2019

Figure 16: Comparison between the different preprocessing similarity score.

Table 20 presents the average percentage score achieved with respect to the manual answer (baseline). The average automatic similarity score performed between 16.5%-27.9% worse than the manual score. The original (unprocessed) similarity score performed between 54%-68.8% worse than the manual score.

	Average Original	Average Automatic
1 - 2018	34.7%	83.3%
6 - 2018	31.2%	76.4%
1 - 2019	45.3%	73.8%
8 - 2019	46.0%	72.1%
Total	40.0%	76.6%

Table 20: The average similarity score for the different text processing with respect to the manual score (baseline).

Table 21 presents the similarity score difference between the comparisons "Original vs. Manual" and "Automatic vs. Manual". The average difference between the original and the manually processed answers is 17.4%, the max score is 32.1%, and the min score is 5.7%. This means that the text processing has the possibility of increasing the similarity by almost one fifth using the simple comparison algorithm. A more advanced comparison algorithm would likely show significantly better results. The similarity score difference between the automatically processed and the manually processed data has an average of 7.2%, with a max score of 25.0% and a min score of -2.7%. There are 2 answers with a lower manual score than an automatic score. Here the machine did a better job than the person. The two negative scores occur in exam 2018.

		Average	**MAX**	**MIN**
1 - 2018	Original vs. Manual	18.8 %	32.1 %	5.7 %
	Automatic vs. Manual	4.9 %	13.3 %	-2.7 %
6 - 2018	Original vs. Manual	16.5 %	23.1 %	9.8 %
	Automatic vs. Manual	6.0%	16.2 %	-1.0 %
1 - 2019	Original vs. Manual	19.8 %	26.9 %	14.2 %
	Automatic vs. Manual	9.9 %	25.0 %	4.6 %
8 - 2019	Original vs. Manual	14.7 %	19.8 %	7.5 %
	Automatic vs. Manual	8.1 %	13.3 %	2.6 %
TOTAL	Original vs. Manual	17.4%	32.1 %	5.7 %
	Automatic vs. Manual	7.2%	25.0 %	-2.7 %

Table 21: The average similarity score difference between "Original and Manual" and "Automatic and Manual" for each answer set.

6.2.5 Discussion and Conclusion

Exercise 1 has tested whether or not the noise reduction pipeline contributed to a better comparison with a reference answer. The trend through all tested answers showed that the original, unprocessed answers performed on average 60% worse than the manual (perfect) text processing. The automatically processed answers performed on average 23.4% worse than the manual text processing. This indicates that noise removal is essential and that the automatically processed answers are doing a good job. Based on these results, the hypothesis is accepted.

The similarity to the reference answers is not particularly impressive. However, this is due to the comparison algorithm, not the preprocessing algorithm. The hypothesis is still valid as we are looking at the trend of improvement, not the absolute values. The best similarity score is question number 1 in exam 2018, with 44.4% (manual) and 36.9% (automatic).

The manually processed answers performed better than the automatic ones and are likely to be due to the lack of certain pipeline steps, namely: synonym word reduction, code-switching, and hyphenation/word splitting.

6.3 Experiment 2

The goal of experiment 2 is to establish which parts of the preprocessing algorithm that has the most significant contribution to noise reduction towards improving the final result. This will be done by creating multiple test preprocessing pipelines where different noise removal steps are present. The same similarity algorithm used in experiment 1 is used to measure the effect of each test. To measure the impact of the noise source, the baseline contains all elements as described in chapter 6.1. The tests will include variations of these elements. All tests are then compared to the baseline. Experiment 2 will use the same answers that were used in experiment 1. However, the analysis will only be done on the automatically preprocessed answers.

Some noise removal steps contribute more to the similarity score than others. Therefore, each of the following tests will state a separate hypothesis. These sub hypotheses are classified as follows: no, little, some, or significant impact.

6.3.1 Method

Based on the dataset analysis in chapter 5, 7 of the noise sources were selected. They were selected based on the amount of noise in the answer or the interesting nature of the noise type. The selected noise sources were: character level, stopwords, Nynorsk to Bokmål, English to Bokmål, term identification, error correction, and normalization. There are 5040 possible pipelines to test with these 7 noise sources. Testing all combinations would not contribute to notable new

information; instead, it would be a time-consuming task. Due to the amount of data, it would not be easy to find correlations. Therefore, only a few hand-picked test combinations were chosen. In total, 5 tests were selected. The combination of noise sources were selected to combine similar noise categories. This was done to minimize the number of tests without disrupting the analysis. By excluding the noise removal in the test, one can find out how the noise removal affects the baseline score. The test combinations are illustrated in table 22.

	Baseline	Test 1	Test 2	Test 3	Test 4	Test 5
Character level	X		X	X	X	X
Stopwords	X	X		X	X	X
Nynorsk - Bokmål	X	X	X		X	X
English - Bokmål	X	X	X		X	X
Term identification	X	X	X	X		X
Error correction	X	X	X	X		X
Normalization	X	X	X	X	X	

Table 22: The test cases done in experiment 2. The different tests contains different steps in the pipeline. In total 5 tests and one baseline with all steps is used.

6.3.1.1 Test 1

Test 1 looks at the noise with regards to character level removal. Character level noise exists in all answers in the dataset and is, therefore, one of the most prominent noise sources. The analysis "punctuation and numbers" from chapter 5.3.5 shows that an answer contains, on average, 18.4 symbol characters for exam 2018 and 25.0 symbol characters for exam 2019. This does not include casing correction, which is also found multiple times in the average answer. The method for handling hyphens ("-") is also one important step in the character level removal due to variations in word contraction and division.

Hypothesis: Character level removal will have a significant contribution to the result.

6.3.1.2 Test 2

Test 2 looks at the noise with regards to stopword removal. The analysis from the stopwords chapter 5.3.1 shows that an average answer from exam 2018 contains 43% stopwords and an average answer from exam 2019 contains 46% stopwords. This is close to half of the answer. By not removing the stopwords, the similarity algorithm would have to compare almost the double amount of tokens.

Hypothesis: Stopword removal will have a significant contribution to the result.

6.3.1.3 Test 3

Test 3 looks at noise with regards to language and regional variations. The analysis of language and regional variations from chapter 5.3.2 shows that for exam 2018, 44.9% of answers contained English words. For exam 2019, 37.8% of answers contained English words. This shows that English words make up a big part of the dataset. In the pipeline, not all of the English words are translated. It is only the words where the Norwegian translation of the word is used in the answers. Nevertheless, it is still a lot of English words that are translated. Usage of Nynorsk is, on the other hand, small, but it does exist. For exam 2018, 2.5% of the answers were written with Nynorsk words, and exam 2019 contained 0.6% of the answers written with Nynorsk words.

There are variations with regards to the use of language and regional variation in the answers. Some answers contain a lot of this noise, and others do not. For the answers containing a lot of this noise type, the removal may significantly impact. One of the answers used in this experiment is written in Nynorsk. The answer is from question 6 in the 2018 exam. For this answer, the translation is likely to impact the result significantly.

Hypothesis: Noise removal from language and regional variations will have a little to some contribution to the result.

6.3.1.4 Test 4

Test 4 looks at noise with regards to error correction and term identification. The analysis of spelling errors from chapter 5.3.3 shows that for exam 2018, 70% of the answers contained spelling errors, and for exam 2019, 84% of the answers contained spelling errors. For each answer, the average amount of spelling errors was 3.8% for exam 2018 and 3.3% for exam 2019. The spelling errors are present in many of the answers. However, they do not contribute to a significant amount of noise.

Hypothesis: Error correction and term identification will have little to no contribution to the result.

6.3.1.5 Test 5

Test 5 looks at noise regarding normalization with the use of lemmatization. The analysis of the ”normalization - lemma” from chapter 5.3.4 shows that the average percentage of words not written in root-form is 14.4% for exam 2018 and 10.5% for exam 2019.

Hypothesis: Word normalization will have some contribution to the result.

6.3.2 Result

Table 23 shows the results of the tests, where the scores are the average percentage score achieved with respect to the baseline.

	Test 1	Test 2	Test 3	Test 4	Test 5
1 - 2018	88.62%	71.96%	98.28%	97.60	60.14%
6 - 2018	91.72%	70.13%	92.50%	99.53%	66.48%
1 - 2019	95.48%	66.95%	100%	95.98%	86.41%
8 - 2019	96.00%	69.92%	100%	98.92%	95.91%

Table 23: The results from the test cases with respect to the baseline.

6.3.3 Discussion and Conclusion

Exercise 2 tested 5 of the most prominent noise sources in the dataset to find their impact within a similarity algorithm. The following paragraphs will discuss the results for each test and accept or reject the test hypothesis.

Test 1 shows that the character level removal only had some impact on the result. The pipeline performed 4.0%-11.38% worse without this step. Question 1 exam 2018 can be seen as somewhat of an outlier as it is a significantly more closed question and requires a shorter answer than the other three. However, the trend shows some but not significant impact on the result. Thus the hypothesis for test 1 can be rejected.

Test 2 shows that the stopword removal had a significant impact on the result. The pipeline performed 28.04%-33.05% worse without this step. This was expected as almost half of the words in the texts were removed. Another reason for the results could be that the stopword list was modified well so that it worked with the dataset. Based on these results, the hypothesis for test 2 can be accepted.

Test 3 shows that the language and regional variations had a low impact on the result. The pipeline performed equal to the baseline for both question sets in exam 2019. This is because the answer set contained no Nynorsk words or English words with a Bokmål translation. It performed 1.72% worse for question 1 in exam 2018 and 7.5% worse for question 6 in exam 2018. The lower percentage score for question 6 in exam 2018 is likely because it contained the only answer of significant length written in Nynorsk in the entire test set. However, for this dataset, the overall impact of translation is low. Thus the hypothesis for test 2 can be accepted.

Test 4 shows that the spelling correction and term identification had a low impact on the result. The pipeline performed 0.47%-4.02% worse without this step. It may be due to the low percentage of errors in each answer or poor performance by the spelling correction software. If the spelling correction software had been more precise, this step would likely have made a more significant impact. Based on these results, the hypothesis for test 4 can be accepted.

Test 5 shows that the lemmatization had a significant impact on the results within the answers in the 2018 exams. Here, the pipeline performed 39.86% and 33.52% worse. Several of the incorrect lemmatizations made by the system lead to more word variations. Because of this, it was surprising that the lemmatization had as big of an impact as it did. The results from the answers in the 2019 exam performed 13.5% and 4.09% worse. These are results that show some impact. Based on these results, the hypothesis for test 5 can be rejected due to the variety of impact.

7 Discussion

In this study, the goal was to research the effect the textual noise contributes in Norwegian exams. Before diving into the discussion, let's look at the research questions one more time.

The research questions for this paper are as follows:

RQ 1: Which are the different noise categories found in Norwegian exams?

RQ 2: Does preprocessing, through noise removal, contribute to a better comparison of Norwegian Computer Science exam answers?

RQ 3: Which parts of a preprocessing pipeline has the greatest impact?

7.1 RQ 1 - Different noise categories

The noise categories found in Norwegian exams are the following: markup, code-switching, wanted terminology, accent, distracting characters, word division and contraction, synonyms, distracting words, spelling errors, grammatical errors, spelling variations, inflections, casing, shortening, and word lengthening. Some of these categories are mentioned under another name in chapter 5 as seen in figure 5. The most prominent noise categories within these exams are spelling variations (both language and regional), distracting words, inflection, technical terms, and synonyms. Other languages' texts are also affected by many of the same noise sources. What is unique about Norwegian exams, when compared to English ones, are spelling variations and technical terms.

Based on the data analysis, the majority of students partaking in the 2018 and 2019 exams write their answers in Norwegian Bokmål, but there exists a handful of student who uses Nynorsk as well. Approximately 2.5% of the answers in 2018 and approximately 0.6% of the answers in 2019. There are also quite a few answers which contain English words, with 44.9% of the answers in 2018 and 37.7% of the answers in 2019 being written partially in English. There are even answers written entirely in English. This is one of the major challenges with developing an Automatic Grading System for the Norwegian language. Whereas English exams normally only have to deal with answers written in English, Norway has 2 recognized written languages. Add in a spattering of English words, and you get a soup that can be challenging to make heads or tails of.

Humans are better suited to compare Norwegian and English words and can differentiate between the two based on context - even when the same word shows up in both languages. When looking specifically at the topic of Computer Science, a big overlap between regular English words and programming language keywords can be observed. Words such as "this", "for", "while", "do", and "loop" can easily be mislabeled as stopwords, and removed, if the grading system hasn't been trained to look for them. Considering answers written in Norwegian, one has to decide which instances of the words "var" and "for" are keywords in a programming language, such as Java,

which are stopwords that should not be considered part of the final preprocessed result. Experiment 1 attempts to answer Research Question 2

7.2 RQ 2 - Noise removal contribution

Preprocessing, through noise removal, does contribute to a better comparison of Norwegian Computer Science exam answers. This can be seen in experiment 1 chapter 6.2, where an unprocessed answer was compared against an automatically and a manually processed answer. The manually processed answer, which has been ”perfectly de-noised”, was used as a baseline for the comparison. The original answers, without any preprocessing, performed the worst, and achieved a 60% worse score. The automatically preprocessed answers took a large step up, but still missed the mark by a good margin, achieving a score that was 23.4% worse. The automatic preprocessing performs well, though it leaves some room for improvement. The most likely reason for this is that the manual preprocessing contains some additional steps in the pipeline, such as code-switching, synonym usage, hyphenation, and an improved spelling correction. With more time, these steps in the manual pipeline could have been developed.

7.3 RQ 3 - Most impactful noise categories

The parts of the preprocessing pipeline that was shown to have the most significant impact were stopword removal and lemmatization. Language and regional variation also had some impact, but only for the answers written in Nynorsk. These can be observed in experiment 2’s 6.3 test 2, test 5, and test 3, respectively.

Stopword removal was the one noise step that was thought to have the most significant impact. This is based on the enormous portion they occupy in the text, with 43%-46%. By not removing stopwords, the pipeline performed 28.04%-33.05% worse. This is one of the ”easiest” noise removal steps to implement because stopword lists are easily accessible for multiple languages. The only work needed is to modify the list to match the dataset it is supposed to be used on.

Lemmatization was one of the noise sources thought to have some impact but showed both very high and very low impact. Test 5 shows that not lemmatizing the words made the system perform 36,69% worse for the 2018 exam and 8,8% worse for the 2019 exam. This is interesting, as the amount of altered words is way less for lemmatization than for stopwords. The average amount of words that were lemmatized per answer was 12,45%.

Spelling variations were one of the noise sources thought to have some impact, which they did. Test 3 shows that the results vary with the amount of Nynorsk in the texts. The one with more Nynorsk performed worse than the other, as the question set containing more Nynorsk performed 7.5% worse, and those with little to no spelling variation performed between 0% and 1.7% worse.

Through the dataset analysis it was discovered which of the types of noise that were present, but the amount of one noise source in a text will not necessarily mean that it will affect the text that much. And based on the information that has been read through it is not clear which noise removal steps that are the most essential to have in place. Looking at lemmatization, we can see improvements of above 30%, even though the lemmatized words only make up about 12% of the total amount. Stopwords, on the other hand, make up almost 50% of the text, even though removing them ”only” improved the performance by about 30%. This goes to show that there is no guarantee for correlation between noise amount and noise removal effect.

Due to the short time frame imposed on master theses, some of the noise categories mentioned in this text have been analysed more thoroughly than others. Several were down-prioritized to limit the scope. This is one of the main reasons as to why the study did not look at the structured part of the answers, e.g. source code - of which there was a lot. For a system to effectively evaluate an answer containing both structured and unstructured text, it first has to separate the two before individually evaluating them using different pipelines. This is because structured texts can have noise sources of their own, such as comments in code, variable names, and an unnecessary amount of parentheses. With the amount of noise that can be present in structured texts, it could very well be a study in and of its own.

One of the intended pipeline steps that had to be completely omitted was synonym replacement. Lack of a good thesaurus for the Norwegian language made it impossible to viably use. The quality of the data, as well as the uncertainty with regards to the origin of the data, resulted in the step providing more headache than value. Not having that step can be a serious detriment to students whose vocabulary might not be as refined as a teacher’s. If this step had been implemented properly, the likelihood of it having a great impact is quite high. Like with lemmatization, the goal of synonym replacement is to reduce the amount of ”equal words”. And as lemmatization made as big of an impact as it did, it is reasonable to believe that synonym replacement would as well.

7.4 What could be done differently?

Knowing the limitations of both the dataset and the NLP resources available for the Norwegian language, this thesis would have had a different focus if it could be restarted today. The thesis would have focused on developing resources for helping creating good question sets, or developing Norwegian word lists for technical terms or synonym usage.

The exam set used in this study was developed with the intention of being manually graded by a human. This was made evident during the initial research as the reference answers were not suitable for automatic grading, and some questions could not be used at all due to their form. The need for better reference answers and questions cannot be ignored, and research into developing a good user interface for teachers to create these seems to be the logical next step in the process.

An alternate next step could be to create a list for technical terms. A lot of the work done within this thesis could have been both easier and better with access to proper word lists. These word lists would ideally be categorized by topic. That way, a system grading Computer Science exams could load a word list for that topic and avoid having, e.g., the word ”htmlen” be corrected to ”himmelen”. The current spelling correction is somewhat inaccurate, as it runs off of a term frequency list created from more general texts. The odds of finding technical terms in news paper articles (which is a common source to use for frequency dictionaries) is not zero, but is statistically improbable. In Iceland, e.g., the Mathematical Society has created a complete dictionary between Icelandic and English with contextual clues for every single mathematical term [40]. If we could create equivalents for various topics in Norway, NLP could leap ahead of where it currently stands. Maybe this is where the National Library or the Language Council could step up to curate these lists and make them freely available. They could sponsor research projects or create ”hackathons” where the participants create quality word lists for set topics. This could help inspire a future surge of interest into the field. When creating these lists, contextual details for word usage should be recorded as well, as terms often carry different meanings for different topics. If you are studying medicine, the word ”terminate” could mean terminating a pregnancy, where as if you’re studying electronics it could mean to terminate a cable, such as when clipping an RJ-45 plug onto the end of an ethernet cable.

There are not only technical term lists that is essential to implement for Norwegian language. There are no open synonym API for Norwegian text. The only automatic method for finding synonyms is scraping public synonym websites. In this thesis, a synonym dictionary was developed from scratch using data from this website, but, as mentioned previously, this dictionary was never put to actual use. Gathering all of the data needed for a single exam took nearly half a day. The web scraper collected synonym data from synonymordboka.no, but due to rate limiting, and the fact that only a single word could be fetched at a time, this took a very long time.

With all this in mind we can clearly see that Norwegian NLP still has a long way to go, and only through collaborative effort is this possible.

8 Conclusion and Further Work

This thesis has researched the impact noise has on automatic grading of Norwegian Computer Science exams. The first step in this study was to identify the noise categories present in the texts, which proved to be quite a few. Within the current scope, there was not enough time, nor enough quality resources, to evaluate the removal of every single noise category in the answers. The results from experiment 1 in chapter 6.2 and experiment 2 in chapter 6.3 clearly shows that the removal of noise does in fact contribute towards a better comparison of student answers and reference answers. Furthermore, the manually preprocessed answers show that there is still room for improvement, and with a more sophisticated comparison algorithm one can hope to achieve the level of accuracy one expects from a human grader.

As expected, the most prominent noise sources were the usage of stopwords and the lack of lemmatization. Language and regional variations also showed some impact, as Nynorsk is not uncommon, and English is everywhere (especially in Computer Science).

Lack of good technical term lists proved to be a real hindrance for both the error correction and the lemmatization steps. It also shines a light on a serious lack of resources within the field of NLP when considering the Norwegian language. Another step that could have had a bigger impact was synonym replacement, which unfortunately had to be scrapped as there is no publically available synonym list — at least not for Norwegian. The upside of this discovery, however, is that it clearly shows which parts of Norwegian NLP that needs more work.

8.1 Further work

As mentioned in the chapter above, there is a need for more NLP resources. One such resource could be a global term database. This database should allow a user to execute a free-text search for a scientific term within a field of study, and then be given a match or a list of probable matches. When exploring said term, the user should be able to see some definition of the term as well as be linked to other languages' version of the same term. By creating this interconnected web of terms, the barrier of entry is lowered significantly for more languages to add their own term lists, thus furthering the field of NLP as a whole and not limiting it to single languages or countries.

It would also be nice if development of a model that could recognize noise in Norwegian answers was possible. But as it stands today, there simply aren't enough open data sets to feasibly create one that would be even remotely adequate for the job. Neither for Computer Science nor for any other topic. Assembling the data sets is not a problem in and of itself, but curating the data that goes into them is. Maybe if all exam answers from both high schools (Videregående skoler) and universities were collected, anonymized, categorized, and made available for people working with NLP then we could see an uptick in not only the quality of work within the field, but also

an increased interest as there was suddenly material to work with. For the 2018 and 2019 exams used in this research there were 164 and 185 answers respectively for each question. Even if every answer to every question was used, and one disregarded the need to split the data into a training part and an evaluation part, the amount is several orders of magnitude too small. Only through sharing of diverse data sets can we see an improvement. The English speaking world is miles and miles ahead of us here. That might be because the majority of the NLP community focus their aim towards English. It's not hard to imagine why, as existing work gives access to vast amounts of libraries, models, data sets, and research. Everything you need to *improve upon* NLP, something that probably motivates a lot of researchers. And as long as researchers continue to focus on English NLP, it will continue to improve. People want to succeed and feel like they make a difference, and as it stands right now, succeeding is much easier when working with the English language. But if we want to reach the same point for the Norwegian language we need to "re-do" all of these old steps, lay the same foundation, and focus our resources away from English.

And just like students might find it hard to write good answers, teachers might find it equally hard to write good questions and reference answers. This was at the very least what was observed in the data sets used in this research, as several reference answers had to be rewritten to allow for good comparison. Another problem that was observed was that teachers, when they write questions, generally wants a lot answered. Often for several different elements within the same question, which often results in a single question containing multiple sub questions. "Answer either this or that"-questions are also questions which are poorly suited for automatic grading as the answer will be ambiguous — does it answer question A or question B? Some questions are also way too open, and could benefit from being narrowed or split into two questions. All of these overhanging "rules" that teachers will have to follow to create good questions leaves room for a tool to step in. Maybe such a tool could be a graphical interface where the teachers could get feedback on their questions as they type them out? One that could show them what the system sees when it analyses the question. Several automatic grading systems have implemented such interfaces before, with C-rater's Alchemist being one of the more known ones.

Bibliography

[1] Khetam Al Sharou, Zhenhao Li and Lucia Specia. 'Towards a Better Understanding of Noise in Natural Language Processing'. In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*. Held Online: INCOMA Ltd., Sept. 2021, pp. 53–62. URL: https://aclanthology.org/2021.ranlp-1.7.

[2] Apertium. *Apertiump*. URL: https://www.apertium.org/index.nob.html#?dir=nno-nob&q=Leken%5C%0A (visited on 9th Apr. 2022).

[3] Sean Bleier. *NLTK's list of english stopwords*. URL: https://gist.github.com/sebleier/554280 (visited on 14th Mar. 2022).

[4] Steven Burrows, Iryna Gurevych and Benno Stein. 'The eras and trends of automatic short answer grading'. In: *International Journal of Artificial Intelligence in Education* 25.1 (2015), pp. 60–117.

[5] *Cosine Similarity – Understanding the math and how it works (with python codes)*. URL: https://www.machinelearningplus.com/nlp/cosine-similarity/ (visited on 22nd May 2022).

[6] Laurie Cutrone, Maiga Chang et al. 'Auto-assessor: computerized assessment system for marking student's short-answers automatically'. In: *2011 IEEE International Conference on Technology for Education*. IEEE. 2011, pp. 81–88.

[7] Hermit Dave. *FrequencyWords*. URL: https://github.com/hermitdave/FrequencyWords/blob/master/content/2018/no/no_full.txt (visited on 10th Apr. 2022).

[8] Gene Diaz. *stopwords-no Public*. URL: https://github.com/stopwords-iso/stopwords-no/blob/master/stopwords-no.txt (visited on 14th Mar. 2022).

[9] Wolf Garbe. *SymSpell*. URL: https://github.com/wolfgarbe/SymSpell (visited on 10th Apr. 2022).

[10] Googletrans. *Googletrans*. URL: https://py-googletrans.readthedocs.io/en/latest/ (visited on 12th Apr. 2022).

[11] Grammarlly. *How Long is a Paragraph?* URL: https://www.grammarly.com/blog/how-long-is-a-paragraph/ (visited on 23rd Mar. 2022).

[12] Sarah Hassan, Aly A Fahmy and Mohammad El-Ramly. 'Automatic short answer scoring based on paragraph embeddings'. In: *International Journal of Advanced Computer Science and Applications* 9.10 (2018), pp. 397–402.

[13] Andrea Horbach, Alexis Palmer and Manfred Pinkal. 'Using the text to evaluate short answers for reading comprehension exercises'. In: *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*. 2013, pp. 286–295.

[14] Richard Klein, Angelo Kyrilov and Mayya Tokman. 'Automated assessment of short free-text responses in computer science using latent semantic analysis'. In: *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. 2011, pp. 158–162.

[15] Vaibhav Kumar et al. 'Machine Learning based Language Modelling of Code Switched Data'. In: *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*. IEEE. 2020, pp. 552–557.

[16] Dhara J Ladani and Nikita P Desai. 'Stopword identification and removal techniques on tc and ir applications: A survey'. In: *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE. 2020, pp. 466–472.

[17] Thomas K Landauer, Peter W Foltz and Darrell Laham. 'An introduction to latent semantic analysis'. In: *Discourse processes* 25.2-3 (1998), pp. 259–284.

[18] Claudia Leacock and Martin Chodorow. 'C-rater: Automated scoring of short-answer questions'. In: *Computers and the Humanities* 37.4 (2003), pp. 389–405.

[19] Jinyu Li et al. 'Robust automatic speech recognition: a bridge to practical applications'. In: (2015).

[20] Teng Li et al. 'Contextual bag-of-words for visual categorization'. In: *IEEE Transactions on Circuits and Systems for Video Technology* 21.4 (2010), pp. 381–392.

[21] Steven Loria. *TextBlob: Simplified Text Processing*. URL: https://textblob.readthedocs.io/en/dev/ (visited on 17th May 2022).

[22] Fetty Fitriyanti Lubis et al. 'Automated Short-Answer Grading using Semantic Similarity based on Word Embedding'. In: (2021).

[23] Thomas S Morton. 'Coreference for NLP applications'. In: *Proceedings of the 38th annual meeting of the association for computational linguistics*. 2000, pp. 173–180.

[24] Nltk. *ESample usage for wordnet*. URL: https://www.nltk.org/howto/wordnet.html (visited on 15th Apr. 2022).

[25] Nltk. *Nltk*. URL: https://www.nltk.org/ (visited on 4th Apr. 2022).

[26] NTNU. *Sensurfrist*. 2022. URL: https://i.ntnu.no/wiki/-/wiki/Norsk/Sensurfrist (visited on 11th May 2022).

[27] Department of Marine Technology NTNU. *IT2810 - Webutvikling*. URL: https://www.ntnu.no/studier/emner/IT2810#tab=omEmnet (visited on 14th Mar. 2022).

[28] Ellis B Page. 'Grading essays by computer: Progress report.' In: *Proceedings of the invitational Conference on Testing Problems*. 1967.

[29] Pypi. *PyDictionary*. URL: https://pypi.org/project/PyDictionary/ (visited on 15th Apr. 2022).

[30] *pyspellchecker*. URL: https://pypi.org/project/pyspellchecker/ (visited on 17th May 2022).

[31] Radim Řehůřek. *Gensim*. URL: https://radimrehurek.com/gensim/ (visited on 4th Apr. 2022).

[32] Rami Al-Rfou. *polyglot*. URL: https://polyglot.readthedocs.io/en/latest/ (visited on 15th Apr. 2022).

[33] Archana Sahu and Plaban Kumar Bhowmick. 'Feature engineering and ensemble-based approach for improving automatic short-answer grading performance'. In: *IEEE Transactions on Learning Technologies* 13.1 (2019), pp. 77–90.

[34] Raheel Siddiqi, Christopher J Harrison and Rosheena Siddiqi. 'Improving teaching and learning through automated short-answer marking'. In: *IEEE Transactions on Learning Technologies* 3.3 (2010), pp. 237–249.

[35] Almås Annabelle Solem. 'Research project for master in automatic grading'. In: (2021).

[36] Beautiful Soup. *BeautifulSoup*. URL: https://beautiful-soup-4.readthedocs.io/en/latest/ (visited on 9th Apr. 2022).

[37] Spacy. *Lemmatizer*. URL: https://spacy.io/api/lemmatizer (visited on 30th Mar. 2022).

[38] Nasjonalbiblioteket - Språkbanken. *Norsk ordbank – bokmål 2005*. URL: https://www.nb.no/sprakbanken/ressurskatalog/oai-nb-no-sbr-5/ (visited on 15th Mar. 2022).

[39] Nasjonalbiblioteket - Språkbanken. *Norsk ordbank – nynorsk 2012*. URL: https://www.nb.no/sprakbanken/ressurskatalog/oai-nb-no-sbr-41/ (visited on 15th Mar. 2022).

[40] *Stæ.is*. URL: https://stae.is/ (visited on 19th May 2022).

[41] Jana Z Sukkarieh, Stephen G Pulman and Nicholas Raikes. 'Automarking: using computational linguistics to score short ,free- text responses'. In: (2003).

[42] Neslihan Süzen. 'From text to meaning: Informational semantic of shot scientific texts'. PhD thesis. 2021.

[43] Oleg Sychev, Anton Anikin and Artem Prokudin. 'Automatic grading and hinting in open-ended text questions'. In: *Cognitive Systems Research* 59 (2020), pp. 264–272.

[44] Kaveh Taghipour, Shahram Khadivi and Jia Xu. 'Parallel corpus refinement as an outlier detection algorithm'. In: *Proceedings of Machine Translation Summit XIII: Papers*. 2011.

[45] TechCommNZ. *How many words make a sentence?* 2016. URL: https://techcomm.nz/Story?Action=View&Story_id=106 (visited on 14th Dec. 2021).

[46] *Text corpus*. URL: https://en.wikipedia.org/wiki/Text_corpus (visited on 31st May 2022).

[47] English Grammar Today. *Punctuation*. URL: https://dictionary.cambridge.org/grammar/british-grammar/punctuation (visited on 24th Mar. 2022).

[48] Keith Vertanen. *Big English Word Lists*. URL: https://www.keithv.com/software/wlist/ (visited on 25th Mar. 2022).

[49] Merriam Webster. *Accent*. URL: https://www.merriam-webster.com/dictionary/accent (visited on 1st Apr. 2022).

[50] Wikipedia. *Edit distance*. URL: https://en.wikipedia.org/wiki/Edit_distance (visited on 14th Apr. 2022).

[51] Wikipedia. *Word embedding*. 2021. URL: https://en.wikipedia.org/wiki/Word_embedding (visited on 14th Dec. 2021).

[52] Wikipedia. *WordNet*. 2022. URL: https://en.wikipedia.org/wiki/WordNet.

[53] Haoran Zhang and Diane Litman. 'Essay Quality Signals as Weak Supervision for Source-based Essay Scoring'. In: *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*. 2021, pp. 85–96.

[54] Wilson Zhu and Yu Sun. 'Automated Essay Scoring System using Multi-Model Machine Learning'. In: *CS & IT Conference Proceedings*. Vol. 10. 12. CS & IT Conference Proceedings. 2020.

Appendix

A 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

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.

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 argumen) 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

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.

16. End to end testing

Forklar hva end to end testing er?

B 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 web- applikasjoner. 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 1

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 2

```
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?

C Experiment 1

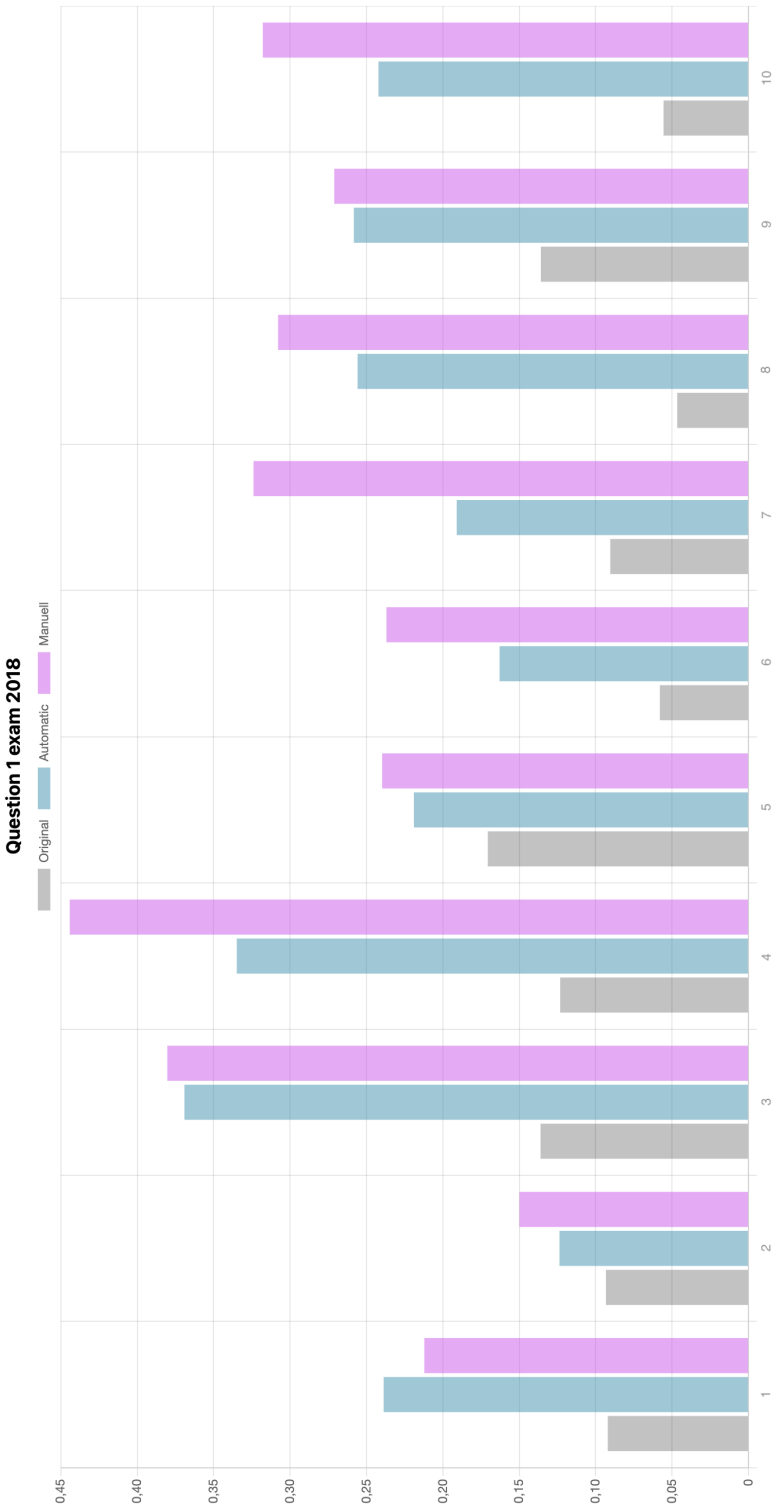


Figure 17: Question 1 exam 2018
Comparison between the different preprocessing similarity score

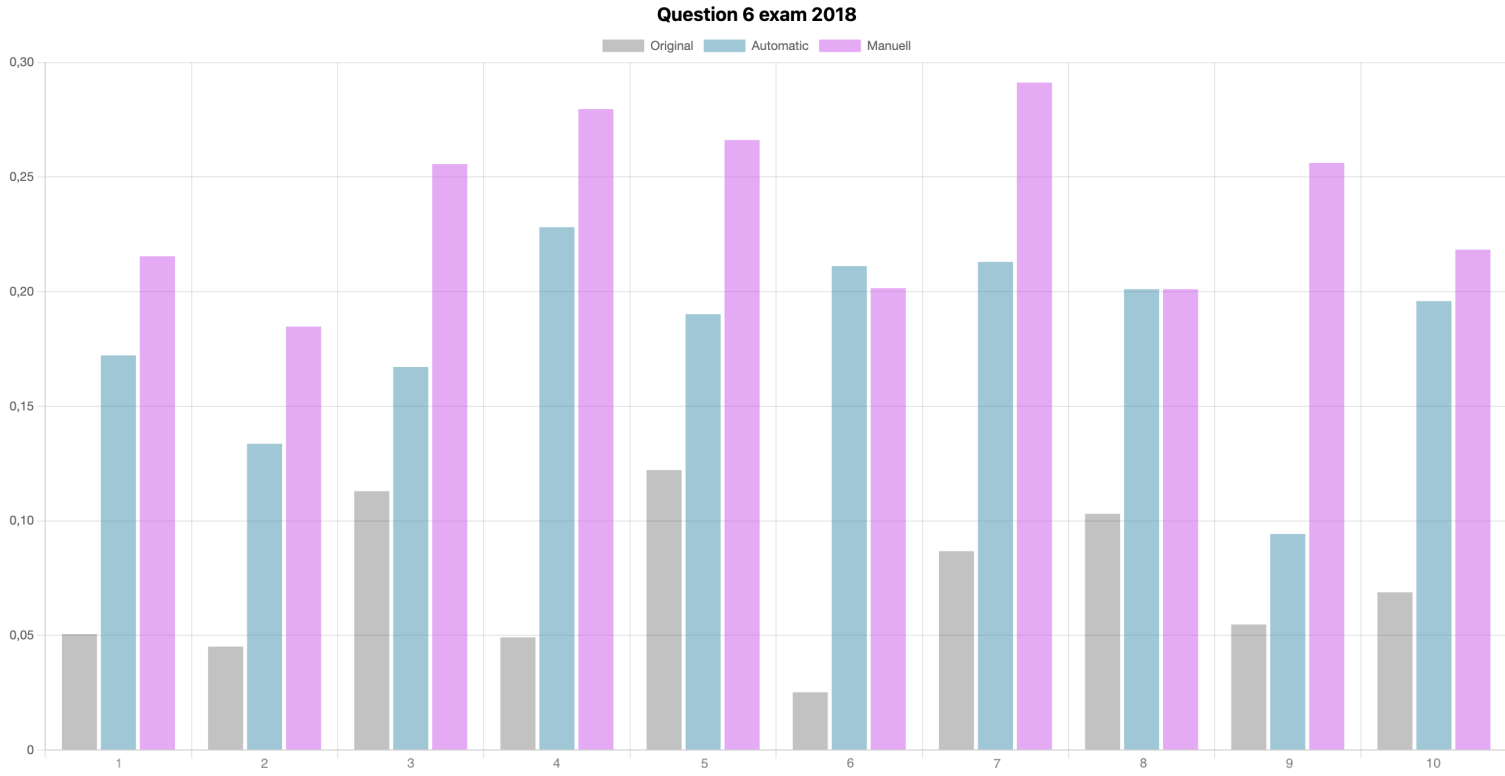


Figure 18: Question 6 exam 2018

Comparison between the different preprocessing similarity score

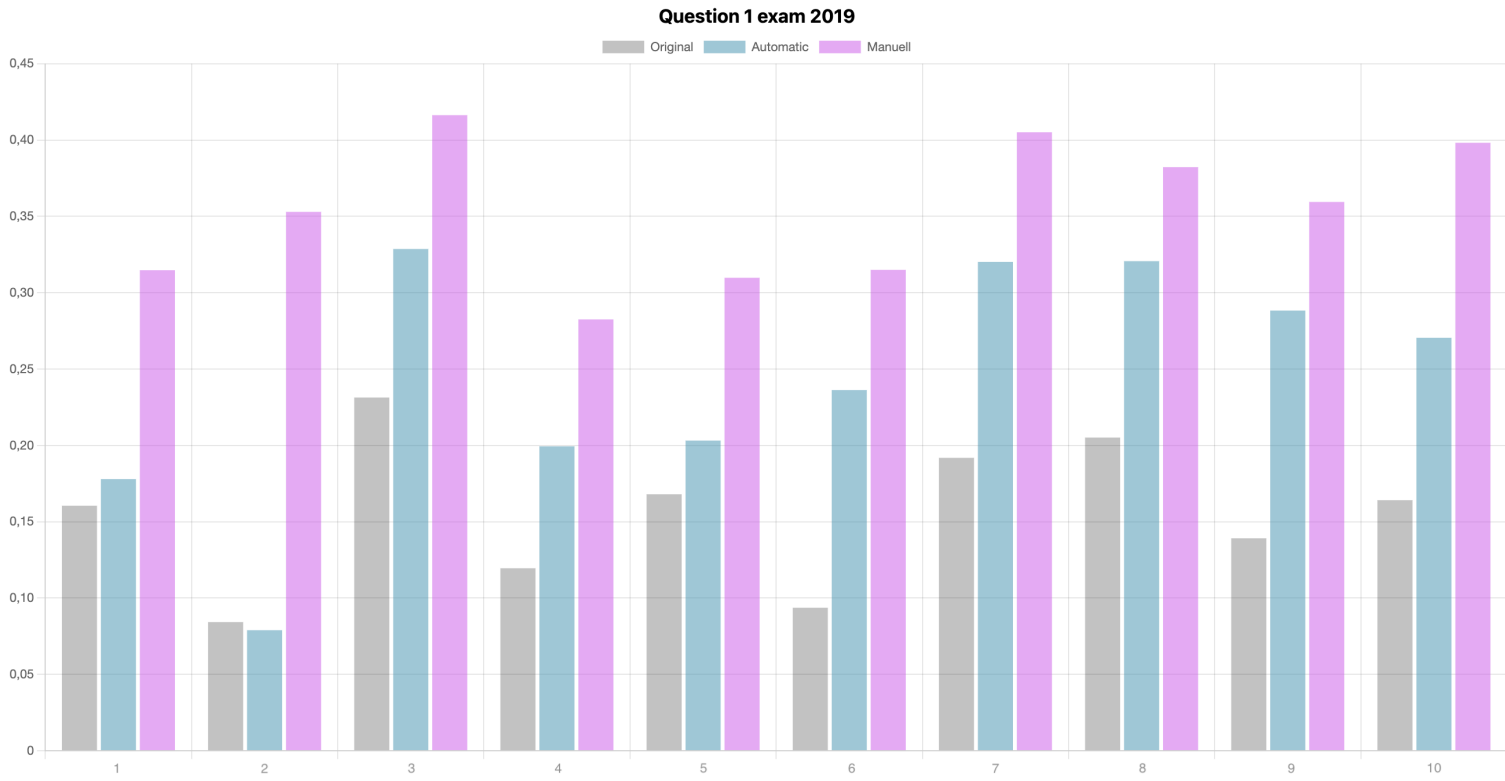


Figure 19: Question 1 exam 2019

Comparison between the different preprocessing similarity score

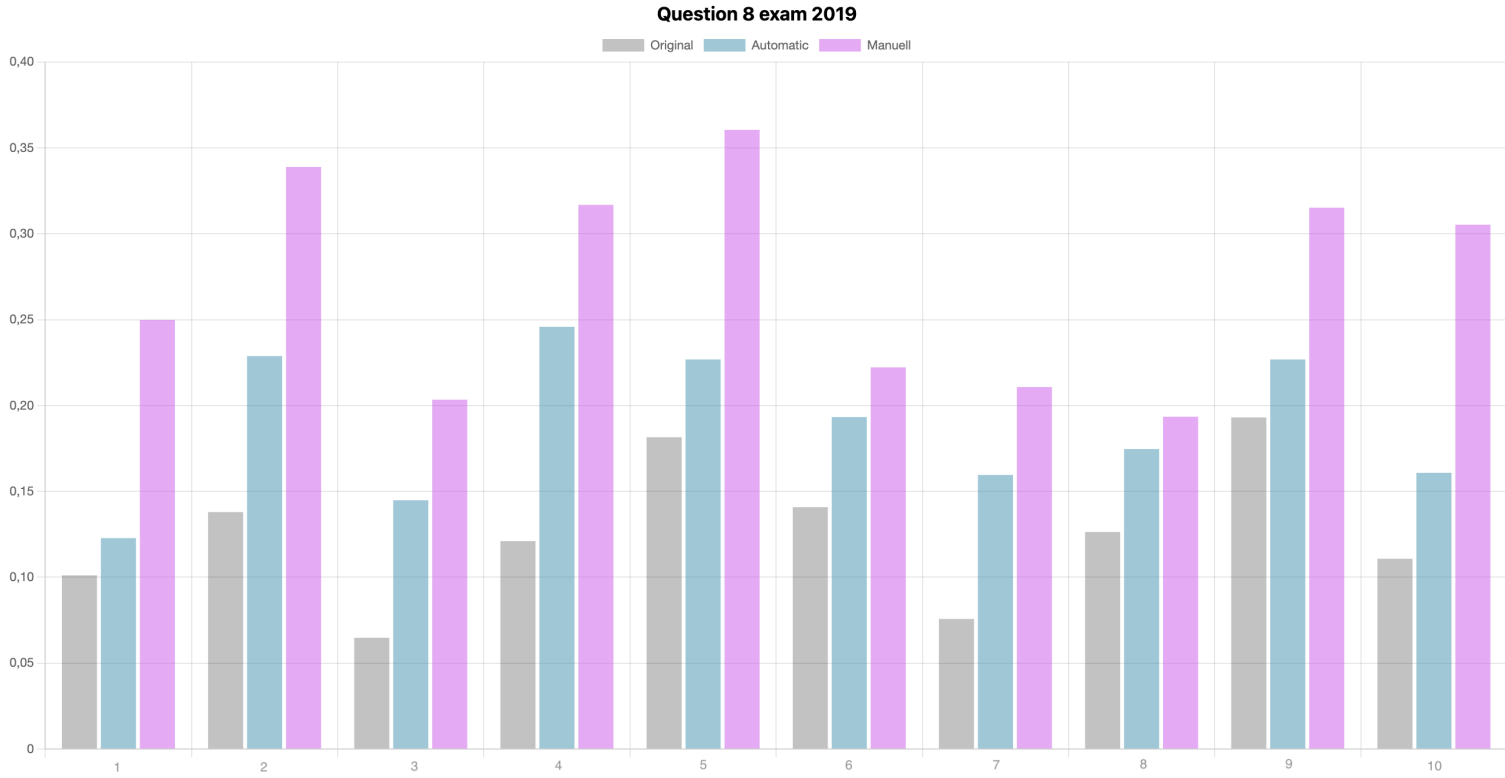
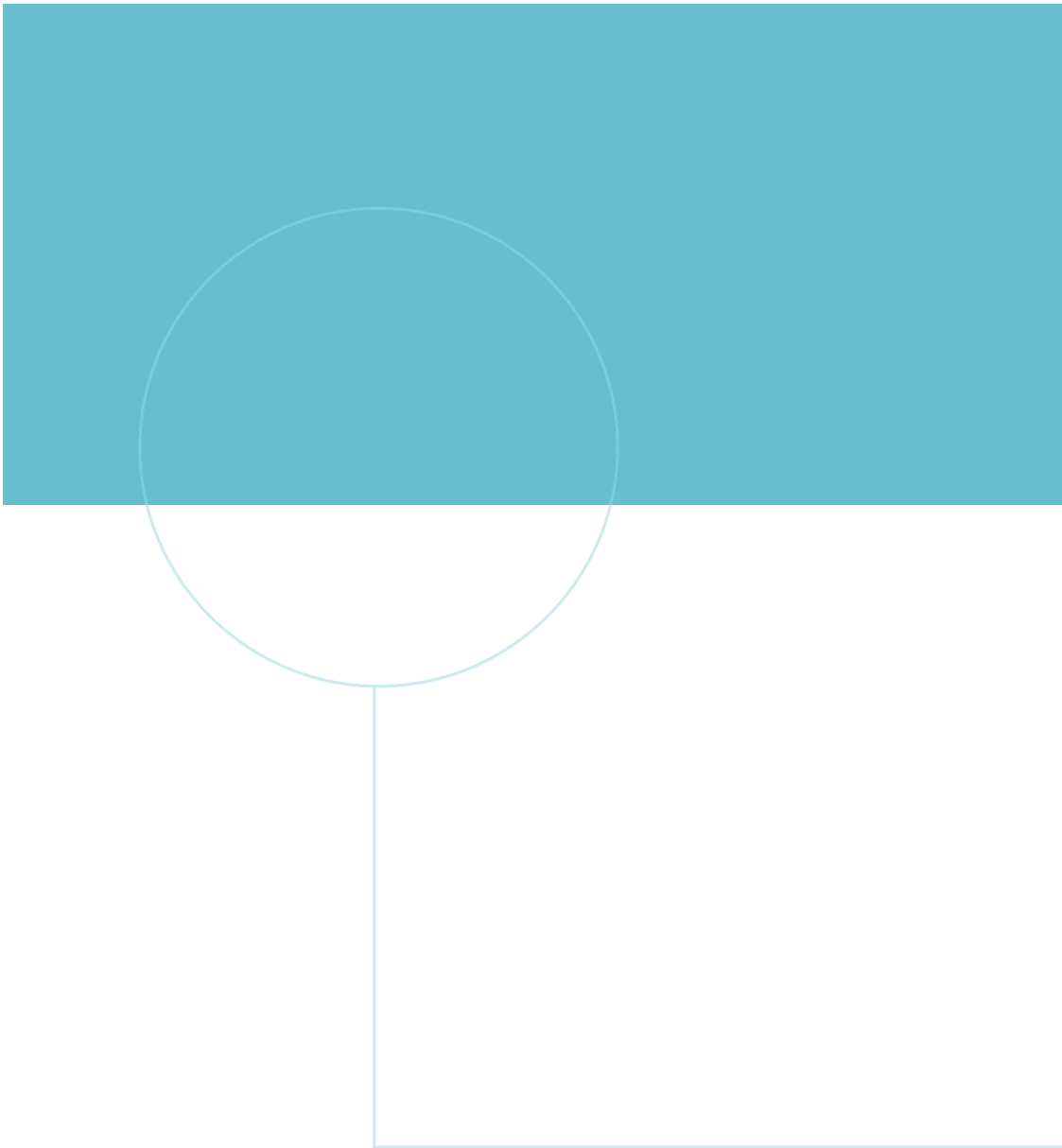


Figure 20: Question 8 exam 2019

Comparison between the different preprocessing similarity score



NTNU

Norwegian University of
Science and Technology