KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

Second Cycle 30 credits

Cluster-assisted Grading

Comparison of different methods for pre-processing, text representation and cluster analysis in cluster-assisted short-text grading

JACOB BÅTH

Stockholm, Sweden 2022

Cluster-assisted Grading

Comparison of different methods for pre-processing, text representation and cluster analysis in cluster-assisted short-text grading

JACOB BÅTH

Master's Programme, Computer Science, 120 credits
Date: June 13, 2022

Supervisor: Johan Boye
Examiner: Olov Engwall
School of Electrical Engineering and Computer Science
Host company: Edaider (data supplied by Natur & Kultur)
Swedish title: Kluster-assisterad rättning
Swedish subtitle: Jämförelse av olika metoder för bearbetning, textrepresentation och klusteranalys i kluster-assisterad rättning

Abstract

School teachers spend approximately 30 percent of their time grading exams and other assessments. With an increasingly digitized education, a research field have been initiated that aims to reduce the time spent on grading by automating it. This is an easy task for multiple-choice questions but much harder for open-ended questions requiring free-text answers, where the latter have shown to be superior for knowledge assessment and learning consolidation. While results in previous work have presented promising results of up to 90 percent grading accuracy, it is still problematic using a system that gives the wrong grade in 10 percent of the cases. This has given rise to a research field focusing on assisting teachers in the grading process, instead of fully replacing them. Cluster analysis has been the most popular tool for this, grouping similar answers together and letting teachers process groups of answers at once, instead of evaluating each question one-at-a-time. This approach has shown evidence to decrease the time spent on grading substantially, however, the methods for performing the clustering vary widely between studies, leaving no apparent methodology choice for real-use implementation. Using several techniques for pre-processing, text representation and choice of clustering algorithm, this work compared various methods for clustering free-text answers by evaluating them on a dataset containing almost 400 000 student answers. The results showed that using all of the tested pre-processing techniques led to the best performance, although the difference to using minimum pre-processing were small. Sentence embeddings were the text representation approach that performed the best, however, it remains to be answered how it should be used when spelling and grammar is part of the assessment, as it lacks the ability to identify such errors. A suitable choice of clustering algorithm is one where the number of clusters can be specified, as determining this automatically proved to be difficult. Teachers can then easily adjust the number of clusters based on their judgement.

Keywords

Machine Learning, Natural Language Processing, Cluster Analysis, Automatic Grading, Automatic Short Answer Grading.

Sammanfattning

Skollärare spenderar ungefär 30 procent av sin tid på rättning av prov och andra bedömningar. I takt med att mer utbildning digitaliseras, försöker forskare hitta sätt att automatisera rättning för att minska den administrativa bördan för lärare. Flervalsfrågor har fördelen att de enkelt kan rättas automatiskt, medan öppet ställda frågor som kräver ett fritt formulerat svar har visat sig vara ett bättre verktyg för att mäta elevers förståelse. Dessa typer av frågor är däremot betydligt svårare att rätta automatiskt, vilket lett till forskning inom automatisk rättning av dessa. Även om tidigare forskning har lyckats uppnå resultat med upp till 90 procents träffsäkerhet, är det fortfarande problematiskt att det blir fel i de resterande 10 procenten av fallen. Detta har lett till forskning som fokuserar på underlätta för lärare i rättningen, istället för att ersätta dem. Klusteranalys har varit det mest populära tillvägagångssättet för att åstadkomma detta, där liknande svar grupperas tillsammans, vilket möjliggör rättning av flera svar samtidigt. Denna metod har visat sig minska rättningstiden signifikant, däremot har metoderna för att göra klusteranalysen varierat brett, vilket gör det svårt att veta hur en implementering i ett verkligt scenario bör se ut. Genom att använda olika tekniker för textbearbetning, textrepresentation och val av klusteralgoritm, jämför detta arbete olika metoder för att klustra fritext-svar, genom att utvärdera dessa på nästan 400 000 riktiga elevsvar. Resultatet visar att mer textbearbetning generellt är bättre, även om skillnaderna är små. Användning av så kallade *sentence embeddings* ledde till bäst resultat när olika tekniker för textrepresentation jämfördes. Däremot har denna teknik svårare att identifiera grammatik- och stavningsfel, hur detta ska hanteras är en fråga för framtida forskning. Ett lämpligt val av klustringsalgoritm är en där antalet kluster kan bestämmas av användaren, då det visat sig svårt att bestämma det automatiskt. Lärare kan då justera antalet kluster ifall det skulle vara för få eller för många.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction and background

A significant part - approximately 30 percent [1] - of school teachers' time is spent on grading exams and assessments. As more education become digitized, researchers have tried to find ways to do automatic grading, decreasing the administrative burden on the teachers. While multiple-choice questions easily can be graded automatically, open-ended questions with free-text answers have shown to be a more valuable tool for measuring the level of understanding among students [2]. This type of question also consolidates learning by forcing students to understand the matter at hand in order to produce the correct answer [3]. Grading free-text answers on the other hand requires significantly more time from teachers as each answer have to be manually evaluated, one-at-a-time.

The last decade, several methods have been published on how to automatically grade free-text answers in what has become known as automatic short answer grading (ASAG) [4] [5] [6]. Prediction accuracy has typically been in the range of 80-90 percent, which is below the 100 percent many teachers and students may require. Unless an automatic grading system is perfect, a teacher might feel urged to review the answers anyway and the students may want to verify that they have received the correct grade.

As a consequence of the imperfect performance in ASAG, a new avenue of research has been growing which instead focuses on assisting teachers in the grading process of free-text answers [7]. The most common approach has been to perform a cluster analysis, which groups similar answers together and then lets teachers handle each group at the same time, instead of evaluating each answer separately. This approach has repeatedly shown to reduce the

time teachers spend on grading [7] [8] [9] and also enables teachers to provide feedback to students in a more efficient manner, as it can be given all at once to a group of students that have answered in a similar way [10]. An additional benefit of assisted grading over a fully automatic one, is that teachers still get to review the answers, letting them improve and adjust their learning, for instance by detecting common misunderstandings [11].

While earlier work has showed that clustering student answers saves time, various approaches for pre-processing, text representation and choice of clustering algorithm have been suggested. Evaluation is usually done on small datasets with 20-30 questions, making it difficult to assess how well a specific approach generalizes to new data with other question characteristics, for instance in terms of answer length and number of answers. If a method for cluster-assisted grading should be implemented in practice, there is no apparent favourable choice of how to do so. This work will evaluate several techniques for pre-processing, text representation and clustering on a large dataset, with the aim to suggest concrete recommendations for real-use implementations.

1.2 Objective and research question

The objective of this work is to find the most suitable method for cluster-assisted grading, ready for implementation as part of an online pedagogical tool. To accomplish this, the following three questions need to be answered:

1. How should each student answer be pre-processed?
2. How should the text be represented numerically?
3. Which clustering algorithm should be used?

These objectives will then help answering the research question: *What are the most suitable techniques for pre-processing, text representation and clustering analysis in assisted grading of short free-text answers?*

1.3 Purpose

This work is carried out together with the online pedagogical tool Edaider*, that offers functionality for teachers to create assessments, such as exams or exercises, where students can answer with multiple-choice alternatives or free

* https://www.edaider.com/

text. While multiple-choice answers can be graded automatically, free-text answers must be processed manually one-at-a-time. The purpose of this work is to examine what pre-processing, text representation and cluster analysis techniques should be used for implementation of a cluster-assisted grading system at Edaider.

1.4 Research methodology

To answer the research question, combinations of techniques for pre-processing, text representation and clustering analysis used in earlier work will be tested on a large dataset with varying types of questions. The evaluation will be done using scoring metrics used in earlier work.

1.5 Delimitations

An important part of assisting teachers in grading is the user interface they will be using when viewing and grading the student answers. This problem is outside the scope of this thesis, but aspects of it will be discussed. While the end purpose of cluster-assisted grading is to facilitate grading for teachers, this work will not test and validate its result by letting teachers use it, various quantitative metrics will instead be used for evaluation.

1.6 Structure of the thesis

Chapter 2 presents the techniques used in the methods examined in this work, as well as related work in more detail. Chapter 3 presents the methodology used to solve the objectives and how to evaluate the results. The result and analysis are then presented in chapter 4 and discussed in chapter 5. Finally, the conclusions are drawn and future work is suggested in chapter 6.

Chapter 2

Background

2.1 Pre-processing and text representations

In order to input text into a clustering algorithm, it has to be represented numerically in what usually is referred to as text or vector embeddings. There are several approaches for creating these embeddings, this section will explain common techniques relevant for this work.

Before the text is transformed, it is usually processed in several ways, with the aim to remove or modify the text so that only essential information is kept. This step is called pre-processing and common techniques include converting all letters to lowercase, punctuation and stopword removal. What determines a stopword may differ but they usually include the most frequently used words in a language, such as "a", "an" and "the". The assumption when using these techniques is that common words, punctuation and letter cases do not contain essential information and can therefore be removed to reduce the size of the text representation as well increasing the chance of finding commonalities between texts.

2.1.1 Bag-of-words (BoW)

As the name suggests, bag-of-words models represent text as a bag of words, ignoring the order in which the word appears but keeping multiplicity, i.e., the number of times every word occurs. A common approach is to create a dictionary of all the unique words in a corpus and represent each word as an element in a $1 \times N$ vector, where N is the number of words in the dictionary. The i:th value representing the word w is then given by the number of times w occurs in a specific text. Alternatively, multiplicity can be ignored using binary

elements, setting each to 1 for all words occurring at least once, otherwise to 0. Bag-of-words have been a common technique in natural language processing and information retrieval for many decades, see for instance [12]. There are also many variations of it, a few of them will be explained in coming subsections.

2.1.2 TF-IDF

TF-IDF is short for term frequency–inverse document frequency, which is a specific type of BoW-model. Only counting every word occurrence or whether a word appears or not, is a sort of term frequency (TF) model. TF-IDF also takes into account the document frequency of each word:

$$IDF(w) = \log \frac{N}{df(w)}$$

where N is the number of documents and $df(w)$ the number of documents w occurs in. TF, the term frequency, can be calculated in various ways but a common one and the one used in this work, is to count the occurrence of each word in a given document. The TF-IDF value for a given word w and document d is then given by

$$TF\text{-}IDF(w, d) = TF(w, d) * IDF(w)$$

where $TF(w, d)$ is the frequency of the word w in document d. The intuition behind TF-IDF compared to only using term frequency, is to reduce the weight of common words that appear in a lot of documents. The technique has been used for several decades, see for instance [13].

2.1.3 N-grams and Chargrams

Bag-of-words models do not take into account the order in which words appear, only their frequency. For instance, the meaning "I do not like vegetables, I like candy", will not capture the fact that "like" appears before vegetables and candy. To do so, we can use so-called bigrams where terms are constructed for all pairs of words appearing in sequence. In this case:

I do

do not

not like

like vegetables

vegetables, I

I like

like candy

A text can then be transformed to a vector where each element corresponds to the frequency of a certain word pair. In the above example however, using bigrams may be problematic as it does not reveal the fact that "do not" appears before "like vegetables". Instead, trigrams may be a better choice which create entities for every triplet of words. The number of words that are combined determines the "n" in n-grams. Bag-of-words, also called unigrams, can then be seen as a special case of n-grams. Additionally, a text representation can be a combination of many n-grams, including both unigrams and bigrams for instance.

Another variation of n-grams is to construct entities of characters instead of words, which often is referred to as char n-grams, or simply chargrams. For instance, the above example using char trigrams would give the entities "I d", " do", "don", and so on.

2.1.4 FastText

A FastText model aims to create vector spaces in which similar words appear close to each other. This is done by representing each word as a real-valued $1 \times N$-dimensional vector, where N usually is in the hundreds. The input for training a FastText model can be a large text corpus, where words are predicted based on its surrounding words appearing in the corpus (called continuous bag-of-words (CBOW)) or where the surrounding words are predicted given the current word (called skip-gram). With the assumption that similar words - in terms of meaning - appear in similar contexts, similar words get similar vector representations and by so placed close to each other in the resulting vector space.

This work used a model specified for Swedish provided by the FastText creators*, which was trained on text from millions of websites (Wikipedia being one of the most common ones) using CBOW. Figure 2.1 displays an example of CBOW, where the word "runs" is predicted based on its surrounding words in the text sequence "the cheetah runs very fast". The

* https://fasttext.cc/docs/en/crawl-vectors.html

surrounding words can be represented as bag-of-words vectors and are then fed into the hidden layers, in which the incoming vectors are summed and multiplied with weight matrices, resulting in an output vector.

When training a model, the aim is to adjust the weights in the hidden layers so that the resulting output vector is as similar as possible to the BOW representation of "runs", called the target. The weight adjustment is done by calculating the difference, called loss, between the output vector and the target, and then computing the gradient in the weight space with respect to this loss, in what is called backpropagation [14]. The weights are then updated in the opposite gradient direction in order to minimize the loss, in what is called gradient descent [15]. This process is then done for all the words in the training corpus and when finished, the learned weights can be used to create word vector embeddings, by multiplying them with the BOW representation of a given word. If the training is successful, similar words would then get similar embeddings. The architecture of the hidden layers may vary; normally several layers are used, i.e., several weight matrices, composing a so-called neural network.

This kind of model resembles word2vec models to a large degree [16]. However, word2vec models uses the surrounding words' BOW representations for prediction, whereas the surrounding words' chargrams are used in FastText. In the example in figure 2.1, this means that the words "the", "cheetah", "very" and "fast" would not be represented as bag-of-words, but as chargrams. With this approach, the internal structure of the words is taken into account, making the model perform better on unseen and misspelled words [17]. Furthermore, in figure 2.1 a window size of 2 is used, meaning that the two previous and following words are used to predict "runs". FastText used a window size of 5, combined with weights which make closer words contribute more to the prediction [17].

Using the FastText model, a text can be represented by averaging all of its word vector embeddings, in what is often called sentence embeddings.

2.2 Cluster analysis

Using any or combinations of the embedding techniques in the previous section, all text data is given a point in some vector space. Cluster analysis is then the task of identifying groups, or *clusters*, of similar data points in this vector space. There are several algorithms to find these clusters, which differ both in the definition of similarity and the way of identifying the clusters. This section will explain three different algorithms that will be used in this paper.

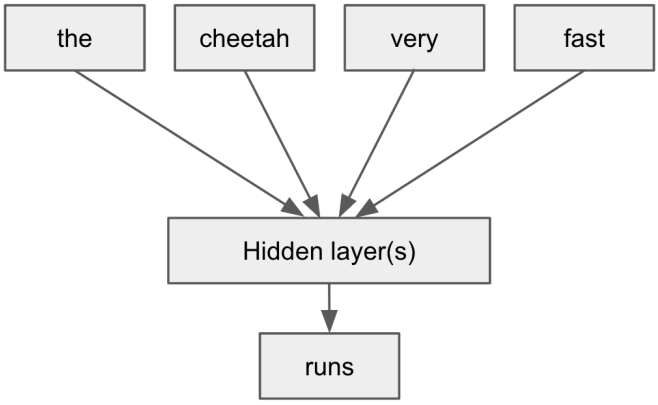

Figure 2.1: Predicting the word "runs" using continuous bag-of-words.

2.2.1 K-means

K-means is a widely used clustering method that have been used for many decades [18]. Given a set of observations $x_1, ..., x_n$, each a d-dimensional vector, and a number of clusters $k \leq n$, k-means iteratively finds the best clusters by comparing distances between the observations. Formally, it minimizes the within-cluster variance between the observations:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

where μ_i is the centroid of the points in cluster S_i. There are various algorithms to determine the cluster for each observation, where one of most common one is Lloyd's algorithm. It alternates between assignment of observations to the closest cluster and then recalculating the centroids (μ). This alternation iterates until the cluster assignments stop changing. The algorithm is not guaranteed to find an optimum and it may find different solutions between runs [19].

2.2.2 Mean shift

The mean shift algorithm aims to find the means, also called modes, that work as the center of each cluster in a set of n observations. It does so by iteratively updating, or "shifting" each observation to the mean in its neighborhood. This neighborhood can be defined by some specific distance or by a weighting

function. Formally, for each data point x a mode $m(x)$ is calculated as

$$m(x) = \frac{\sum_{i=1}^{n} w(i)x_i}{\sum_{i=1}^{n} w(i)}$$

, assigning $x \leftarrow m(x)$ is then called shifting. $w(i)$ is usually a kernel function, for example the Gaussian kernel $e^{-c\|x_i - x\|^2}$ where c is a bandwidth parameter which dictates how concentrated the weighting should be around observation x; as c increases, the weighting decreases faster around x. This is the only parameter to the algorithm and must be set beforehand, the number of clusters will then be determined automatically. However, it must be set with care, a too narrow one may cause each observation getting its own cluster, a too large one will result in a single large cluster containing all data points [20].

2.2.3 Agglomerative clustering

Agglomerative clustering is a hierarchical clustering method. It starts by placing each observation in a separate cluster and then successively merges them based on some linkage criteria. A common criterion is called Ward, which minimizes the sum of squared distances within all clusters [21]. The algorithm stops when reaching a beforehand specified number of clusters or distance threshold. The method is hierarchical as clusters are merged successively, so that a merge can be seen as a combination of many *subclusters* into one larger one.

2.2.4 Determining the number of clusters

The mean shift method does not require the number of clusters to be set beforehand. In K-means, this must be set and also in agglomerative clustering unless a distance threshold is specified. A popular method for determining the most suitable number of clusters is called the elbow method [22]. It starts with trying out $K = 2$ clusters, K is then incremented with 1 up to any value. For each K, the clusters generated by the algorithm (e.g. k-means) are scored by some cost or score function J. The result may then be plotted as in figure 2.2 with the hope that a *elbow* would appear, in this case at $K = 4$ clusters. For greater K the decrease in J is so small that it is usually not worth creating an additional cluster for. If J is a scoring function so that higher values are desirable, the shape of the curve will be different - typically drastic increase that flattens out after a certain point - and the objective is then to find the *knee*. Except for the different shape, all principles of the method remain the same.

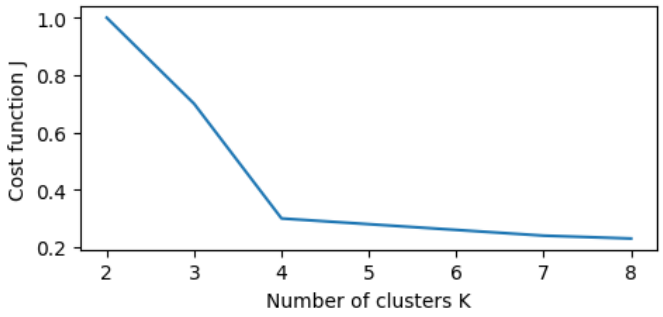

Figure 2.2: Identifying the most suitable number of clusters with the elbow method. In this case, 4 clusters is the best.

While it is easy to identify the elbow in 2.2, it may not always be that obvious, for instance if the decrease in J is constant. In this case, clustering algorithms such as mean shift may seem favorable as the number of clusters is determined automatically. This algorithm on the other hand then requires a bandwidth parameter to be set, which can be as tricky as finding an elbow.

2.2.5 Cluster evaluation

Cluster analysis evaluation is no trivial task. While there are numerous commonly used metrics, most have some drawback by failing to capture some aspect, see for example [23] for an in-depth comparison between different evaluation metrics. This work will use metrics that have been used in related work, with the following explanations.

For each cluster j, class i and number of data points n, purity is calculated as

$$Purity = \frac{1}{n}\sum_{j}\max_{i} n_{ij}. \tag{2.1}$$

This metric measures to what extent every cluster contains a single class, for instance a single grade. It ranges from 0 to 1, where a higher value is desirable. With the same notation, entropy is given by

$$Entropy = \sum_{j}\frac{n_j}{n}(-\sum_{i}\frac{n_{ij}}{n_j}\log_2\frac{n_{ij}}{n_j}) \tag{2.2}$$

where lower values are desirable. A drawback with purity and entropy is that they reach their maximum value if assigning each data point to a specific cluster, which is why it is important to take the number of clusters into

consideration as well.

Silhouette value is another metric that do not requires the true class of each data point. It measures how similar and separated each data point is to its own cluster and other clusters respectively. It ranges from -1 to 1, where a high value signifies well-formed clusters. The silhouette value for each data point i is defined as

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

where a is the mean distance between i and all other data points in its cluster j and b is the mean distance between i and all other data points in the most similar cluster to j, also called the neighbouring cluster. A single silhouette value, using any distance metric, can then be obtained by taking the mean of all silhouette values in all the clusters.

2.3 Related work

2.3.1 Automatic Short Answer Grading

Automatic Short Answer Grading (ASAG), also referred to as Automatic Short Answer Scoring (ASAS), is a research area where the objective is to automatically grade short free-text answers to a question. The context is usually that a teacher has asked a question and then receives tens or hundreds of answers to this question. As opposed to multiple choice questions where the grading procedure can be fully automated, free-text answers need to be analyzed by the teacher one-at-a-time. This process can be very time consuming, urging the need of assisting teachers in this process.

In the last 20 years there have been several promising studies in this research area [4] of automatically grading short free-text answers [5] [6]. Short answers typically range from a single word to a few sentences and a usual approach to determine the grade is to compare an answer to a given correct one by the teacher. Before the comparison, the answer is pre-processed and then transformed into a vector embedding.

2.3.2 Assisted short answer grading

While previous research has presented promising results in ASAG, no model is 100 percent correct, forcing teachers to double-check corrections for a fair evaluation. This has motivated studies where the objective is to assist the teacher in the grading process instead of fully replacing it [7] [8].

Powergrading, presented in [7], clustered similar student answers together and suggested a grade for each cluster. This lets teachers evaluate similar answers at the same time, see the suggested grading and adjust it if necessary. This work used data from 20 questions, each having around 600 answers. For pre-processing, the authors of this work removed stopwords and words in the answers that also appeared in the question, in what they called *question demoting*. The answers were then represented as a combination of features, including TF-IDF vectors and the difference in length between the answer and correct answer. The clustering was then made with the k-medoids algorithm, which is similar to k-means except that each cluster is represented by the most central data point, instead of the mean of all data points in given cluster [24]. To grade the answers, a model was trained to classify the similarity between pairs of answers. This type of grading requires manually annotated data for how similar two answers are and the classifier could not generalize to make predictions for questions that were not used for training. The result of this work showed that the proposed method managed to reduce the number of actions needed by the teacher to correct all answers from 600 (the number of answers) to 10-100 actions, depending on the type of question. The clustering also managed to group similar wrong answers - cases where the students were confused in the same way. The grading model failed however to detect correct phrases that were paraphrases of the correct answer, suggesting that synonym detection may be useful.

In a follow-up work, the same authors designed and tested a user interface for teachers to grade answers and give feedback to students. This user interface used the powergrading algorithm for clustering the answers and they found that the teachers were able to grade and give feedback substantially faster, as well as developing an overview of common understandings and misunderstandings among the students [9].

In a more recent study [8], a similar method was presented but with a different clustering method (k-means) and a lower number of answers (around 30) on each question. Pre-processing techniques used included punctuation removal, stopword removal, lemmatization, question demoting and removal of words only occurring in 1-2 answers. The answers were represented as normalized bag-of-words vectors and the elbow method was used to determine a suitable number of clusters. To grade the answers, a regression model was trained based on similarity between answers and the correct answer. This worked well for some questions but poorly on others. As in [7], the grading model in this work also had to be trained anew for every question. The authors found that the grade (1-5) correlated with how many of the words in the correct

answer were used in the student answer. However, the importance of the words in the correct answer varied and the authors suggested that this should be marked by the teacher before doing the automatic grading. Furthermore, synonym detection was mentioned as an improvement suggestion. Despite varying performance of the grading, the clustering was tested with teachers and were able to reduce the workload significantly.

In another study [25], the authors compared different clustering methods and found that most are effective when used for short answers, but result in poor performance when the answers are several sentences long. This work used n-grams, and chargrams in their word representations, where the latter mainly accounts for spelling errors. As previous mentioned work, this report also built models that are trained on specific questions, without the purpose or ability to generalize to other questions.

In [26], question demoting together with keyword highlighting were used, where the latter are binary features for whether certain keywords appear in the answer or not. These keywords are manually handpicked, but in general they are the most important words in the correct answer. Furthermore, they tested the clustering by letting teachers grade answers with and without the clustering tool. They found that using clustering, the teachers were able to grade answers faster than without the tool. It is not clear how the tool was designed and exactly how much faster grading it resulted in. To evaluate the clustering, purity and entropy score were used.

2.3.3 The benefits of assisted grading

While fully automatic grading in theory has the ability to eliminate all time teachers spend on grading, recent models have presented accuracy of 80-90 percent [4] [5] [6], which is below the 100 percent teachers may strive for. In addition, there are other benefits of actually reviewing the students' answers.

Feedback plays an important role in education and is known to facilitate learning [10]. For students, there is a great value of knowing what was wrong in their answer and how they can improve, instead of only be given the fact that it was wrong. In automatic grading, there is currently no clear way of how to provide extensive feedback, as this requires the teacher to actually review the answers. Using clustering on the other hand, does not only enable feedback-giving, but also speeds up the process as the same feedback can be given to whole clusters of similar answers, instead of being provided individually.

By reading the students' answers, the teacher can also learn about their knowledge level and common misconceptions. This is a part of *formative*

assessment, where assessments are used by teachers to improve their learning methods [11]. This process may be improved by clustering answers, as it is easier to detect patterns of common understandings and misunderstandings among the students when similar answers are grouped together.

Many, if not most students, have some time felt they have been unfairly graded. In [27], both students and teachers reported that grading is often a "hodgepodge" combination of achievement, attitude and effort, where achievement should be the only aspect that is examined. It is also possible that equal answers get different grades due to inattention by the teacher or that multiple teachers have been involved in the grading process, where they are not fully aligned on grading criteria. In clustered-assisted grading, similar answers are grouped together and a teacher can grade all these answers in one action, disregarding the student's attitude and effort behind every answer. This may also increase the probability of assigning the same grade to equal answers.

Chapter 3

Method

3.1 Software and hardware used

Python was used as programming language for implementing all tested methods and the evaluation script. Scikit-learn [28] was also widely used as well as some other libraries that will be mentioned where used. The evaluation script was run on Amazon Web Services using a machine with 64 virtual CPUs (vCPU). The iteration was split into 64 different processes, each running on a separate vCPU, in order to speed up the process.

3.2 Data

The data for this work was obtained from Natur & Kultur*, who offers an online learning tool with various functionality where students submit answers to teacher-created questions. The dataset contains a total of 384 915 student answers, each accompanied with the question asked by the teacher, correct answer suggested by the teacher and the teacher-supplied correction of the answer (correct or incorrect). The number of unique questions were 6514, meaning an average of 59 answers per question, with a minimum of 10 and maximum of 551. 90 percent of the answer were marked as correct and the remaining as incorrect. While this imbalance could be remedied by only selecting the appropriate questions, it would lower the data quantity substantially. In addition, the data is not used for training of a particular model that would risk being biased using imbalanced data. The data is merely used for evaluating different methods and there is therefore a value in keeping the

* https://www.nok.se

real distribution of correct and incorrect answers.

The questions were of the type "Instuderingsfrågor", roughly translating to study questions, meaning that the students typically have answered the questions out of classroom, for instance before an upcoming test. As this work intends to generalize to other use cases as well, i.e., to be used in-classroom, it would be favourable with answers obtained from multiple environments. However, the questions are not on a specific form for "Instuderingsfrågor" and could just as well have been asked in a classroom or on a test.

The nature of the questions varies a lot; some ask for some specific information and other ask to explain and elaborate on something. The answers are however not longer than a couple of sentences. The students that answered the questions are in 1-12th grade, which typically means 6-19 years of age. No personal data is included in the dataset.

3.3 Implementation

3.3.1 Pre-processing

This section will describe all the different pre-processing configurations and techniques used for this work.

Nothing / minimum pre-process

This setting was used to process as little as possible before embedding the text as vectors. All punctuation and casing were kept, only removing tab and newline characters.

Basic

This step involved removing punctuation and everything else that were not letters or digits, where the latter were replaced with "<num>". All remaining letters were then converted to lowercase.

Stopword removal

Stopwords are usually very common words such as "a", "an", and "the". The implementation simply looked at each word, tried to find it in a given list of stopwords and then removed the word if found in the list, otherwise it was kept. For this work, a stopword list from Spacy* was used, containing 386 Swedish

* https://spacy.io

words.

Lemmatization

In a similar manner to stopword removal, this step looked at each word and tried to find its lemma, or base form, in a list. If found, the original word was exchanged with the lemma. A lemma list from Spacy was used.

Question demoting

This step removed words in the answer that also occurred in the question. For each word, if its original form or its lemma occurred in the question's original form or lemmatized form, it was removed.

Spelling correction

This step used a spell correction package called Hunspell*. Each word was searched for in a dictionary; if found, the word was kept. If not found, similar words were suggested by the Hunspell package, ranked by the degree of similarity. If there were no suggestions, happening when the word was not similar to any word in the dictionary, the word was kept. Each suggestion was then searched for in any of the other student answers, in order to favour these suggestions. This was done as some suggestions clearly were not the intended word by the student as it was completely out of context. If the word however occurred in other student answers, it is likely to be the intended word. If none of the suggestions were found in the other student answers, the word was replaced with the most similar suggestion. Similarity was only based on the used letters and not the meaning of the words.

The pre-processing techniques were then combined into the following different configurations:

- Nothing: Split sentences by space (keep punctuation, casing, etc)
- Basic: Nothing + lowercasing, remove punctuation, replace numbers with <num>
- Basic + stopword removal

* http://hunspell.github.io

- Basic + lemmatization
- Basic + question demoting
- Basic + spell correction
- All: Basic + stopword removal, lemmatization, question demoting, spell correction
- All - stopword removal
- All - lemmatization
- All - question demoting
- All - spell correction

3.3.2 Text representations

Six different text representation settings were used: bag-of-words (BoW), TF-IDF, chargrams, BoW n-grams, TF-IDF n-grams and sentence embeddings. Scikit-learn [28] was used for all settings except for sentence embeddings, as it only requires the student answers as input to create the desired representation. When n-grams were used, a range of 1-3 was used, meaning unigram, bigrams and trigrams were included. When chargrams were used, a range of 3-5 letters were used. These ranges were suggested by Zesch et al. [25] and optimizing them is outside the scope of this work.

Fasttext [17] was used to create the sentence embeddings. It takes a text sequence and creates a word embedding for each word, i.e., a vector with dimensionality $1 \times N$. All the word vectors are then averaged, resulting in a $1 \times N$-dimensional vector representation for the given student answer. N was initially set to 300, but this was not practically doable given the hardware used and amount of data that had to be processed. Instead, $N = 50$ was used.

3.3.3 Vector normalization

The data was evaluated with and without the use of normalization. Without normalization however, the results were substantially worse and the clustering algorithms often failed as the vector space spanned by the text representations became too large. Because of this, the results without normalization were discarded and not further analyzed.

When normalization was used, it was done on a per-feature basis, making the values of each feature have zero mean and unit variance, where each element in the vector embeddings represents a feature. This is exemplified in 3.1, where three bag-of-words vectors for a dictionary consisting of the words "the", "cat" and "dog" are shown. In the upper table, the actual counts of the words are displayed while the lower table shows the values after normalization. The normalization is done by first re-scaling all the values for "the" so that the mean is zero and variance is one, then doing the same for "cat", and so on. With this approach, the order of magnitude of each feature is ignored, letting them contribute equally to the result. For instance, without normalization, the distances between the BOW vectors would mostly be determined by their values in the "the" column, as the differences between the vectors for this word are greater than for the other words. With normalization however, the values for "the" are of the same magnitude as "cat" and "dog".

	the	cat	dog
BOW 1	20	1	0
BOW 2	10	1	1
BOW 3	3	2	1

	the	cat	dog
BOW 1	1.29	-0.71	-1.41
BOW 2	-0.14	-0.71	0.71
BOW 3	-1.15	1.41	0.71

Table 3.1: BOW representations without (upper table) and with (lower table) normalization.

3.3.4 Clustering

Three different clustering algorithms were used: k-means, meanshift and agglomerative clustering. K-means were used as it has been used in earlier work [7], agglomerative clustering as it enables hierarchical clustering - which could be useful when designing a user interface for cluster-assisted grading - and meanshift was used as it automatically determines the number of clusters. All three algorithms were implemented with Scikit-learn [28].

As meanshift does not require the number of clusters to be specified, it only takes the text representations as input. For k-means and agglomerative

clustering however, the elbow method was used to find a suitable cluster number. As the algorithm were to be tested on thousands of questions, it was not possible to visually identify an elbow for each. Instead, the elbow/knee was found automatically by picking the cluster number at which the curvature of the cost/score curve was the greatest. This was implemented with Kneed* using silhouette value as score function.

3.4 Evaluation

The different techniques for pre-processing, text representation and clustering were then used in a grid search, evaluating all data for each possible combination of the three steps. Each combination and question resulted in an assignment of each student answer to a specific cluster. As the teacher's correct answer also was included in the clustering, it was also assigned to a cluster. All the answers in this cluster were then marked as correct and all the remaining answers as incorrect. The clusters were then evaluated using purity, entropy (defined in equation 2.1 and 2.2) and "actions left", as all of these have been used in previous related work [7] [26].

Actions left were calculated as the number of needed changes to reach the same correction status given in the original dataset. This was done by summing for each cluster (with correct and incorrect here referring to whether the assigned grade is the same as in the dataset, see figure 3.1 for visual explanation):

- If all answers graded correctly -> 0 actions required. See upper left cluster in figure 3.1.
- If all answers graded incorrectly -> 1 actions required (as all grades are switched in one action). See upper right cluster in figure 3.1.
- If mixed grading, switch the n incorrect ones. If n is greater than half the cluster size N, switch the whole cluster (one action), then switch the $N - n$ incorrect gradings -> $min(n, N - n + 1)$ actions required. See bottom clusters in figure 3.1.

* https://github.com/arvkevi/kneed

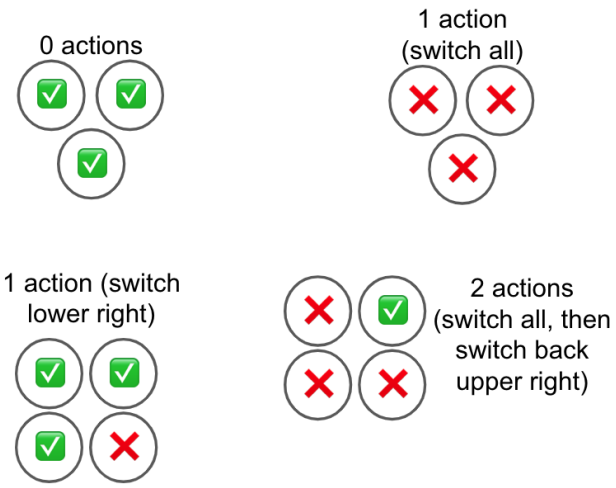

Figure 3.1: Four clusters where each circle represents an answer and the green/red mark whether it's assigned the same grade as in the dataset. In this example, a total of four actions are required to achieve perfect grading. Note that this procedure requires binary grading.

Entropy and actions left increases with the number of answers n, they were therefore normalized by dividing by n. In addition to the presented evaluation metrics, the number of clusters were taken into consideration in the analysis as well as the number of answers n and average length of the student answers m, by reporting how the evaluation metrics depend on n an m. An example of clustered answers was also presented to give a sense of how meaningful the clustering is and whether it is useful.

The reason for using multiple evaluation metrics is that none of them are perfect, as they may fail to measure a certain aspect of the cluster analysis - for instance, purity and entropy would be maximized by splitting creating a separate cluster for each data point. The use of multiple metrics is therefore a way to increase the validity of the results as more aspects of the clustering is taken into account.

Chapter 4

Results and Analysis

The grid search tested 11 pre-processing techniques, 6 text representations and 3 clustering algorithms over 6514 questions, resulting in $11 * 6 * 3 * 6514 = 1289772$ values for the evaluation scores purity, entropy and actions left. This chapter will present averages and distributions of these values, grouped by either clustering algorithm, text representation or pre-processing technique.

4.1 Clustering algorithms

Figure 4.1 shows the scores for the three different clustering algorithms. While the purity scores are very similar, there is a large difference in the other scores, with meanshift standing out. Agglomerative and k-means clustering both have a bit more than 2 in average number of clusters, while this number is around 15 for meanshift.

As a result of yielding more clusters, meanshift has lower entropy. While this is favorable in general, it is not a perfect measure as 0 entropy would be achieved by letting each answer have its own cluster. Figure 4.2 shows the distribution of answers per cluster for each clustering algorithm. Meanshift seems to create clusters only containing a few answers, regardless the total number of answers. K-means and agglomerative clustering on the other hand, have a varying number of answers per cluster as it often splits all the answers into two clusters only, as seen in figure 4.1.

Finally, actions left is considerably higher for meanshift, which also is a result of the higher cluster number as more clusters have to be processed.

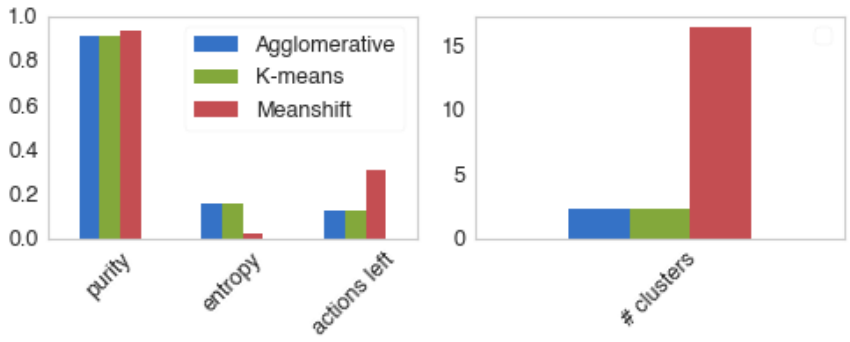

Figure 4.1: Scores for different clustering algorithms. The relation between the evaluation types are not of interest, merely the difference between the algorithms for each score. Note that entropy and actions left are normalized by the number of answers.

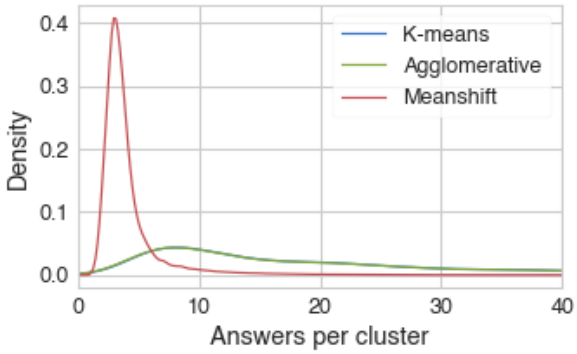

Figure 4.2: Distribution of answers per cluster for each algorithm. K-means and agglomerative clustering are nearly identical (the line for k-means is hidden by the line for agglomerative clustering).

4.2 Text representations

Table 4.1 shows the average number of clusters for each text representation technique. As meanshift had a significantly higher number of clusters, its values are separated to the right. While most representation techniques have similar values, sentence embeddings (Sentembed in the table) stand out. It produces relatively more clusters for agglomerative and k-means clustering, while significantly lower values than the others for meanshift

clustering. Figure 4.3 and 4.4 show the distribution of number of clusters for agglomerative and k-means clustering and meanshift clustering respectively, with a curve for each text representation. All representations are almost identical, except for sentence embedding (blue) that differs. For agglomerative and k-means, it is clear that for all representation techniques except for sentence embeddings, only two clusters are created in most of the cases. Sentence embeddings on the other hand have a wider distribution. In the meanshift case, sentence embeddings tend to yield a lower number of clusters than the other representation techniques.

Representation	# clusters	Representation	# clusters
BoW	2.13	BoW	18.0
BoW + ngram	2.12	BoW + ngram	18.9
Chargram	2.14	Chargram	17.8
Sentembed	3.10	Sentembed	8.5
TF-IDF	2.15	TF-IDF	17.1
TF-IDF + ngram	2.11	TF-IDF + ngram	18.3

Table 4.1: Average number of clusters per data representation. As agglomerative and k-means clustering had nearly identical results, their values are averaged in the left table. The right table shows values for meanshift.

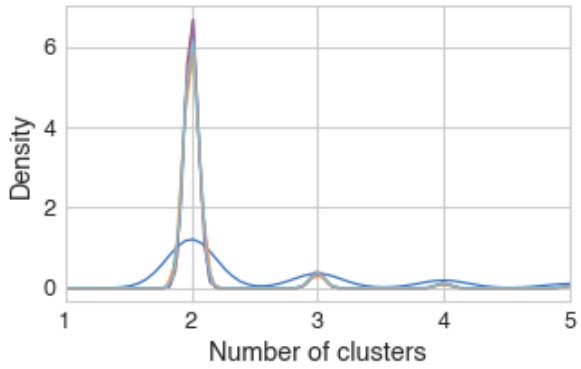

Figure 4.3: Distribution of number of clusters per text representation for agglomerative and k-means clustering. All representations are almost identical, except for sentence embedding (blue) that stands out. The graph is cut at 5 number of clusters to display the difference between text representations.

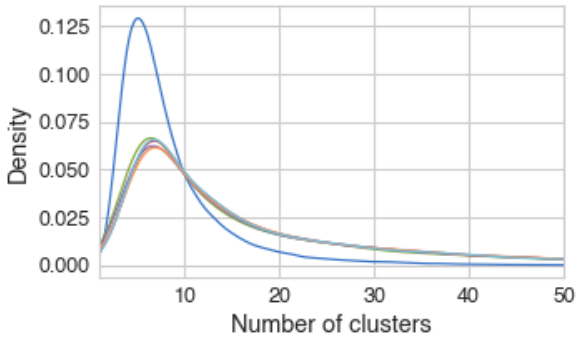

Figure 4.4: Distribution of number of clusters per text representation for meanshift clustering. All representations are almost identical, except for sentence embedding (blue) that stands out.

Viewing the other scores for each representation in table 4.2, it is clear that sentence embeddings continue to differ from the very similar results of the other techniques. As entropy to a large degree is determined by the number of clusters, it is a likely reason for sentence embedding's lower value for agglomerative and k-means clustering and higher value in meanshift, compared to the other techniques. Regarding purity and actions left, sentence embeddings perform better for all clustering algorithms.

4.3 Pre-processing

As meanshift clustering and sentence embeddings have shown to yield significantly different results, they will be separated when comparing pre-processing techniques. Table 4.4 and 4.5 display scores for different pre-processing configurations, using k-means/agglomerative clustering and meanshift respectively. The tables are then divided into results using sentence embeddings, or any of the other representation techniques.

In general, purity and entropy improve with more pre-processing procedures, although the differences being small. Actions left are very similar, while the number of clusters increase with more pre-processing procedures. When sentence embeddings are used, stopword removal is the most determining factor, with a clear increase in number of clusters when it is used and a clear decrease when it is not. Question demoting is the second most determining factor for number of clusters. When sentence embeddings are not used, the difference in number of clusters is much smaller.

Representation	Purity	Entropy	Actions left
BoW	0.910	0.167	0.123
BoW + ngram	0.908	0.170	0.125
Chargram	0.909	0.167	0.124
Sentembed	0.925	0.114	0.123
TF-IDF	0.910	0.165	0.123
TF-IDF + ngram	0.908	0.170	0.124

Representation	Purity	Entropy	Actions left
BoW	0.935	0.021	0.325
BoW + ngram	0.929	0.023	0.345
Chargram	0.933	0.022	0.327
Sentembed	0.946	0.034	0.194
TF-IDF	0.936	0.025	0.307
TF-IDF + ngram	0.930	0.024	0.334

Table 4.2: Scores for each data representation. The upper table show values for agglomerative and k-means clustering, the lower table shows values for meanshift. Note that entropy and actions left are normalized by the number of answers.

Pre-processing setting	Explanation
Nothing	Split sentences by space (keep punctuation, casing, etc)
Basic	Nothing + lowercasing, remove punctuation, replace numbers with <num>
+Stop	Basic + stopword removal
+Lemma	Basic + lemmatization
+Demote	Basic + question demoting
+Spell	Basic + spell correction
All	Basic + Stop, Lemma, Demote, Spell
-Stop	All - stopword removal
-Lemma	All - lemmatization
-Demote	All - question demoting
-Spell	All - spell correction

Table 4.3: Explanation of pre-processing configuration/setting names in table 4.5 and 4.4.

	Purity	Entropy	Actions left	# clusters
Nothing	0.908	0.169	0.124	2.1
Basic	0.909	0.168	0.124	2.12
+Stop	0.909	0.168	0.124	2.13
+Lemma	0.909	0.168	0.124	2.13
+Demote	0.909	0.168	0.124	2.11
+Spell	0.909	0.168	0.124	2.13
-Stop	0.91	0.167	0.124	2.13
-Lemma	0.91	0.167	0.124	2.14
-Demote	0.909	0.167	0.124	2.14
-Spell	0.91	0.167	0.124	2.14
All	0.91	0.166	0.123	2.15

	Purity	Entropy	Actions left	# clusters
Nothing	0.922	0.124	0.122	2.79
Basic	0.923	0.123	0.122	2.82
+Stop	0.925	0.111	0.124	3.2
+Lemma	0.923	0.123	0.121	2.81
+Demote	0.924	0.118	0.123	3
+Spell	0.923	0.122	0.122	2.85
-Stop	0.924	0.117	0.122	3.03
-Lemma	0.927	0.103	0.125	3.48
-Demote	0.926	0.11	0.124	3.24
-Spell	0.927	0.106	0.125	3.41
All	0.928	0.103	0.125	3.49

Table 4.4: Scores for different pre-processing configurations using k-means and agglomerative clustering. Sentence embeddings are used in the lower table, all the other representation techniques in the upper one.

4.4 Varying question characteristics

This section will present how different question characteristics affect the evaluation scores. As meanshift and sentence embeddings have shown to lead to the largest differences in score, the results will be split up into four groups based on these settings.

	Purity	Entropy	Actions left	# clusters
Nothing	0.928	0.0262	0.323	17.7
Basic	0.931	0.0239	0.329	18.1
+Stop	0.932	0.0227	0.335	18.6
+Lemma	0.932	0.0237	0.328	18.1
+Demote	0.932	0.0232	0.329	18.2
+Spell	0.932	0.0237	0.325	17.7
-Stop	0.933	0.0227	0.322	17.6
-Lemma	0.934	0.0218	0.326	17.9
-Demote	0.933	0.0219	0.326	17.7
-Spell	0.934	0.0214	0.335	18.7
All	0.935	0.0213	0.323	17.6

	Purity	Entropy	Actions left	# clusters
Nothing	0.943	0.0363	0.183	7.54
Basic	0.944	0.0355	0.188	7.94
+Stop	0.948	0.0324	0.196	8.74
+Lemma	0.945	0.0351	0.188	8
+Demote	0.945	0.0347	0.196	8.64
+Spell	0.944	0.0363	0.186	7.74
-Stop	0.946	0.0345	0.193	8.41
-Lemma	0.948	0.0314	0.202	9.2
-Demote	0.948	0.0319	0.193	8.5
-Spell	0.949	0.0299	0.208	9.79
All	0.949	0.0305	0.203	9.31

Table 4.5: Scores for different pre-processing configurations using meanshift clustering. Sentence embeddings are used in the lower table, all the other representation techniques in the upper one.

4.4.1 Varying answer length

Figure 4.5 shows how the scores change depending on the average number of words in the answers. Although all settings start off on different levels, they all follow similar patterns in purity.

Entropy-wise, meanshift have a lower increase in the beginning than when using agglomerative and k-means clustering. This is likely due to the fact that meanshift in general creates more clusters, which lowers entropy.

Looking at actions left, using sentence embeddings or not creates different

forms on the curves. When in use, actions left increases in the beginning, drops and then flattens out. Not using sentence embeddings gives a curve that increases in the beginning and then flattens out.

For actions left, k-means and agglomerative clustering do not change very much. Meanshift, when not using sentence embeddings, increases drastically initially and then flattens out on a relatively high level. Using meanshift with sentence embeddings on the other hand results in a lower and somewhat more stable value.

As for all these metrics, it is important to consider that they are affected by the number of clusters, making it difficult to compare methods if they differ significantly on this aspect. For instance, not using sentence embeddings results in a very low value for actions left, but at the same time the worst values in purity and entropy. This would in practice mean that the teacher would not need to change a lot in the suggested grading, but still has to analyse the clusters to identify the answers that are labeled incorrectly. Meanshift with sentence embeddings on the other hand has higher actions left but also better purity. The teacher may then have more clusters to review, but it is possible that it will take less time in the end as the answers are more similar in each cluster.

4.4.2 Varying number of answers

Figure 4.6 displays how the scores depend on the number of student answers. Meanshift when not using sentence embeddings have a slightly different pattern than the other settings purity-wise. Entropy decreases for an increasing number of answers when using meanshift and increases when using agglomerative and k-means clustering. Actions left decreases for all settings and then flattens out, where meanshift and sentence embeddings have the largest drop.

4.5 Clustered example question

Table 4.6 shows an example of clustered answers to the question "Vad händer med koldioxiden som frigörs vid cellandningen?" (translation: "What happens to the carbon dioxide released by the cellular respiration?"). The answers are in Swedish, but non-Swedish readers can still get a sense of the similarity between the answers based on their structure. This example used all pre-processing techniques, sentence embeddings and k-means clustering. The clusters are separated by horizontal lines and the answers within each cluster are ordered by silhouette values, in order to present similar answers within each cluster close to each other. The answers graded as incorrect by the teacher

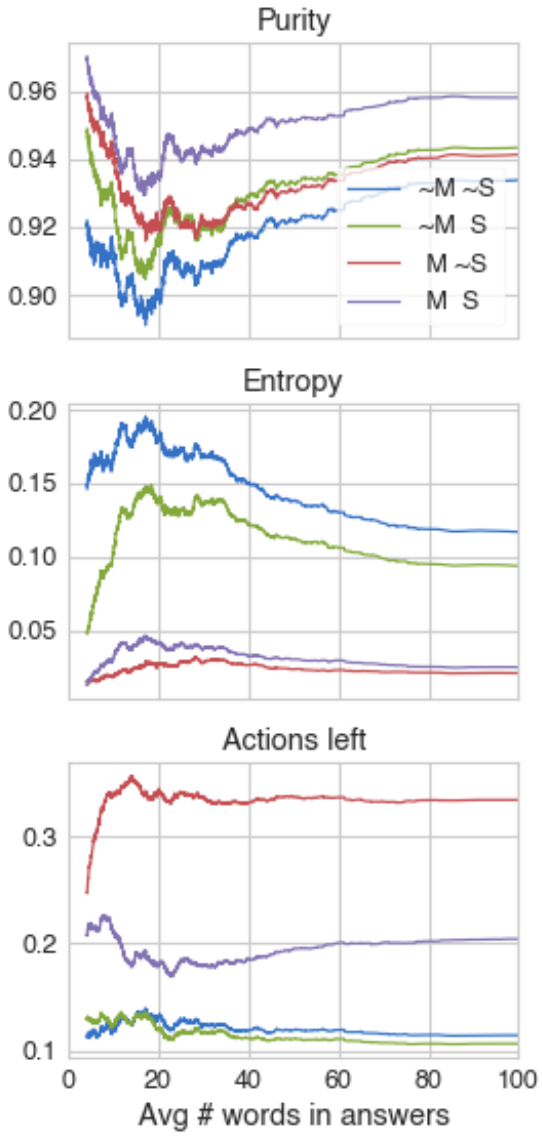

Figure 4.5: Scores (500-rolling mean) depending on the number of words in the teacher-given correct answer. The different curves correspond to different settings. M: using meanshift; ∼M: using agglomerative and k-means clustering; S: using sentence embeddings; ∼S: using the other aforementioned representation techniques.

are marked with red text, the correct-graded answers are black. The correct answer supplied by the teacher was "Den tas upp av växterna.", which would

Figure 4.6: Scores (500-rolling mean) depending on the number of student answers. The different curves correspond to different settings. M: using meanshift; ∼M: using agglomerative and k-means clustering; S: using sentence embeddings; ∼S: using the other aforementioned representation techniques.

suggest the grading "correct" for the third cluster with the most answers. The actions needed to reach perfect grading for this example would be to identify

the correct answers in cluster 2 and 4. While this cluster analysis performed perfect purity and entropy, it is not meant to be used as any sort of validation, merely to give a visual sense of the clustering.

Looking at the clusters, the first one seems to have grouped short and, in some cases, completely wrong answers together - for instance "sdfgh". The second cluster contains two almost identical answers, suggesting that two students have been working together or possibly copied the answer from a textbook. The third and largest cluster contains the most answers. Thanks to the sorting by silhouette values, it is fairly easy to overview most of the answers as similar ones are placed together. It is also worth noting that spelling errors are ignored, as "växter" and "vexter" are grouped together as well as "koldioxid" and "koldioxin". The remaining clusters only contain one answer each, probably as these differed too much from all the other answers.

CLUSTER 1
- syre
- koldioxiden flygger till ett träd och blir till syre med hjälp av bladen på trädet
- Släpps ut i luften
- sdfgh
- Byggsten och energi
- människor och djur andas ut och in den.
CLUSTER 2
- Cellandning eller respiration sker i alla levande organismer, även växter. I mörker sker bara cellandningen i växten. Då använder växten syre och glukos från fotosyntesen. Då frigörs bland annat energi och koldioxid.
- Cellandning eller respiration sker i alla levande organismer, även växter. I mörker sker bara cellandningen i växten. Då använder växten syre och glukos från fotosyntesen.
CLUSTER 3 (Suggested grading: correct)
- Den tas upp av växterna.
- Det tas upp av växterna.
- Den tas upp av andra växter
- det tas upp av växterna
- den tas upp av västerna
- växterna tar upp det
- den tas upp av växterna
- den tas up av vexter
- Den tas upp av växterna.
- växter tar upp den.
- växten tar det
- Överbliven koldioxid tas upp av växterna.
- överbliven koldioxid tas upp av växterna
- överblivna koldioxid tas upp av växterna
- vid nedbrytning av döda djur/organsimser så frigörs koldioxid och det tas upp av växterna
- växterna tar upp det för att kunna fortsätta fotosyntesen.
- Växter andas in koldioxid.
- Växterna andas in koldioxin
- växterna använder det som energi
- Den åker ut ur kroppen och ut till naturen där växterna använder et för fotosyntes
- Människor och djur andas ut koldioxiden som frigörs. Vid nedbrytning av döda organismer frigörs koldioxid, som växter behöver till sin fotosyntes.
CLUSTER 4
- växter och andra levande organismer som innehåller pigment tar upp kol i form av koldioxid från luften och med hjälp av vatten och energi från solen omvandlas kolet till kolhydrater i fotosyntesen.
CLUSTER 5
- I cellernas mitokondrier sker den så kallade cellandningen som är en process där lagrad energi i födan frigörs. Energin finns bunden i sockermolekyler när den når cellen. Med hjälp av enzymer bryts sockermolekylerna ned till vatten och koldioxid. ... Förbränningen av druvsocker eller glukos sker hela tiden i våra celler.
CLUSTER 6
- Det som händer är att där bränsle regeraar med syre kallas förbränningar därför händer det att cellandnigen kallas för förbränningen.

Table 4.6: Clustered answers to the question "Vad händer med koldioxiden som frigörs vid cellandningen?" (translation: "What happens to the carbon dioxide released by the cellular respiration?") using all pre-processing techniques, sentence embeddings and k-means clustering. The clusters are separated by the horizontal lines. Red text indicates that the answer was labeled as incorrect by a teacher in the dataset.

Chapter 5

Discussion

5.1 Pre-processing

None of the tested pre-processing configurations showed a significant worse or better result in any of the measured scores. As the minimum pre-processing configuration (only splitting the text by space) performed similar to the others, a possible implementation may be to have this as standard and then let teachers select extra pre-processing techniques, for instance to correct spelling. This option is useful as spelling often can be a part of the assessment and misspelled words should therefore be easy to detect by the teacher. Lemmatization faces a similar dilemma as it changes each word to its base form. If the conjugation of the word is an important part of the assessment, lemmatizing words should be avoided. Question demoting can also be problematic for certain questions; for instance, using question demoting on a question like "Is X or Y correct?" is inappropriate as X and Y would be removed from every answer. In practice, it is not certain that there exists an optimal configuration, instead it depends on the nature and purpose of the question.

5.2 Word representations

Sentence embeddings were the only word representation technique that stood out from the rest. Except for higher entropy when used with meanshift, it had the best 4.2 score on all evaluation metrics. While this is promising, it is important noting that sentence embeddings are worse at detecting spelling errors and other possibly important details, as the representation of a word and a misspelled version of the word may be minuscule, causing them to be treated as identical. While it may be desirable for some questions, it is not so when

spelling is part of the assessment. In this case, sentence embeddings could be combined with some other technique, such as bag-of-words. Furthermore, this work used a 1×50-dimensional sentence embedding which is relatively small, optimizing this may be a topic for future research.

5.3 Clustering

Using k-means and agglomerative clustering often resulted in only two clusters, whereas meanshift ranged from a few up to around 50. As with pre-processing, the optimal number may be very dependent on the nature of the question. For a question like "Is X or Y correct?", 2 clusters seem appropriate but in other cases, it is not certain it is helpful at all if their answers vary too much within each cluster. Having too many clusters, possibly with only one answer in some clusters, would not either be of much help as it is similar to going through the answers one-at-a-time.

While the elbow method used for k-means and agglomerative clustering could have been further optimized to avoid too few clusters, another possible approach is to find a cluster size that works well in general. Basu et al.[7] did not automatically decide the number of clusters, but instead tested various numbers to find a suitable number and then used this for all questions. They concluded that 10 clusters were suitable for questions of around 600 answers. Within these clusters they also created subclusters which is easily done when using agglomerative clustering. In practice, a possible approach is to start with a suitable number based on the number of answers, then letting the teacher adjust the number of clusters based on their judgement, simply by re-running the clustering algorithm with the new cluster size. This is also possible implementation-wise, as the text representations are kept and re-running the clustering algorithms usually can be done in less than a second using a standard machine.

Meanshift on the other hand does not have the option to specify the number of clusters. In this regard, k-means and agglomerative clustering may be more appropriate unless the number of clusters always is set to a suitable level. This was not the case when using other text representations than sentence embeddings, as the number of clusters grew very large.

5.4 Practical use

The end goal for an assisted grading system is to let teachers spend less time on grading and providing feedback. A crucial part for this is to design an algorithm that creates meaningful clusters, but just as important is the user interface that displays the clusters and its answers to the teacher. There needs to be an easy way to grade and give feedback to each cluster, but also overview each individual answer within the clusters, as concluded by Basu et al. [9] when they tested a user interface for cluster-assisted grading. While creating a user interface goes beyond the scope of this work, the chosen clustering method may impact how easily overviewed each cluster is. If answers are grouped solely based on its content while ignoring text format, it may be hard to overview several answers if the sentences vary too much in form. In the example in table 4.6, the top half of the third cluster is fairly easy to overview as the sentences are on very similar form. Had not the answers been ordered by silhouette value and displayed in random order, it may be much harder to overview, even though the answers contain the same information. Stopwords and lemmatization can in this sense be valuable, as common words and their conjugation may contain information about a text's structure.

5.5 Validity

A key determining factor for the evaluation metrics used in this work is the number of clusters. It is therefore difficult to compare performance between methods when the number of clusters varies significantly. Moreover, the end purpose of this work is not to gain the best values on these evaluation metrics, but rather to let teachers grade and give feedback to students in a more efficient manner. Validating promising methods by letting teachers use them in practice is therefore a natural next step.

The data used for evaluation had a high fraction of correct answers, making it easy to obtain high purity and low entropy. For instance, if 90 percent of the answers are correct, 0.9 in purity is already achieved using a single cluster. While it would be possible to only evaluate on data with lower fraction of correct answers, this would drastically decrease the data size.

Finally, it is not given from the data in exactly what environments the students have answered the questions. The example in table 4.6 have two longer answers that are almost identical, suggesting that the students have had access to some learning material or have been sitting together when answering

the questions. If the questions are answered during an exam, it is possible that the answers would vary more, making it harder to group similar answers together.

5.6 Ethical aspects

Grades are an important aspect of education and something that can affect students in several ways, from their wellbeing to whether or not they will be accepted to a school or job. It is therefore crucial that grading is done in a correct and fair way, why this work has chosen to facilitate grading, instead of fully automating it. Nevertheless, using cluster-assisted grading will let teachers process students in groups instead of as individuals. Should the clustering not be perfect, it is possible that some grading will be done incorrectly if the teacher is not observant enough. Another undesired scenario is that teachers rely too heavily on the clusters and fail to see answers that are very different, but still would be graded as correct if observed separately. While it is common to have an established way for students to ask for re-evaluation, it can be a time-consuming process for both teachers and students, possibly leading to more time spent on grading than if the teacher would make sure to grade correctly from the beginning.

Chapter 6

Conclusions and Future work

6.1 Conclusions

Although the differences are small, the experiments show that using more pre-processing techniques in general leads to better performance and a higher number of clusters. Stopword removal and question demoting led to the largest increase in number of clusters, suggesting that these techniques help distinguish different answers the most. While all the tested pre-processing techniques have shown to boost performance, it is important to have in mind that they may cause undesired results for some types of questions. For instance, question demoting should not be used on questions such as "Is it X or Y?", as X and Y would be removed from all answers.

In terms of word representations, sentence embeddings stood out from the other methods with better performance on all metrics. However, it is important to consider the fact that it may fail to detect spelling or grammatical errors. Sentence embeddings may therefore be replaced or combined with some other technique, such as bag-of-words, if spelling and grammar is part of the assessment.

When comparing clustering algorithms, the large difference lay between meanshift versus k-means and agglomerative clustering, where meanshift in general led to more clusters and better performance in terms of purity and entropy. However, while having a very large number of clusters may lead to high purity and low entropy, it could possibly increase the grading time as more clusters have to be reviewed. As determining the number of clusters automatically is difficult, k-means and agglomerative clustering may be more useful as the number is given beforehand and can easily be adjusted if a teacher thinks there are too few or too many. Implementation-wise, this is possible as

the clustering algorithms can be re-executed in less than a second, enabling quick experimentation by the user until meaningful clusters are formed.

6.2 Future work

This work has compared different methods for cluster-assisted grading using quantitative measures, which serve as a proxy for the real purpose, namely the time it takes for teachers to grade and give feedback. A natural next step is therefore to test the suggested method choices in this work in practice with real teachers. This also requires a user interface for displaying the clusters, which also is a possible future avenue of work.

Optimizing the suggested methods is also something left undone. Sentence embeddings showed promising performance while using a relatively low embedding size. It is likely that further performance improvements can be made by altering this size. Furthermore, how sentence embeddings should be used for questions where spelling and grammar is assessed, remains to be answered.

References

[1] O. Mason and I. Grove-Stephensen, "Automated free text marking with Paperless School," Jan. 2002, publisher: Loughborough University. [Online]. Available: https://repository.lboro.ac.uk/articles/conference_contribution/Automated_free_text_marking_with_Paperless_School/9490169/1

[2] J. H. McMillan, "Secondary Teachers' Classroom Assessment and Grading Practices," *Educational Measurement: Issues and Practice*, vol. 20, no. 1, pp. 20–32, 2001. doi: 10.1111/j.1745-3992.2001.tb00055.x _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1745-3992.2001.tb00055.x. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1745-3992.2001.tb00055.x

[3] J. D. Karpicke and H. L. Roediger, "The Critical Importance of Retrieval for Learning," *Science*, vol. 319, no. 5865, pp. 966–968, Feb. 2008. doi: 10.1126/science.1152408 Publisher: American Association for the Advancement of Science. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1152408

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

[5] M. A. Sultan, C. Salazar, and T. Sumner, "Fast and Easy Short Answer Grading with High Accuracy," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, Jun.

2016. doi: 10.18653/v1/N16-1123 pp. 1070–1075. [Online]. Available: https://aclanthology.org/N16-1123

[6] W. H. Gomaa and A. A. Fahmy, "Ans2vec: A Scoring System for Short Answers," in *The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019)*, ser. Advances in Intelligent Systems and Computing, A. E. Hassanien, A. T. Azar, T. Gaber, R. Bhatnagar, and M. F. Tolba, Eds. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-14118-9_59. ISBN 978-3-030-14118-9 pp. 586–595.

[7] S. Basu, C. Jacobs, and L. Vanderwende, "Powergrading: a Clustering Approach to Amplify Human Effort for Short Answer Grading," *Transactions of the Association for Computational Linguistics*, vol. 1, pp. 391–402, Oct. 2013. doi: 10.1162/tacl_a_00236. [Online]. Available: https://doi.org/10.1162/tacl_a_00236

[8] N. Süzen, A. N. Gorban, J. Levesley, and E. M. Mirkes, "Automatic short answer grading and feedback using text mining methods," *Procedia Computer Science*, vol. 169, pp. 726–743, Jan. 2020. doi: 10.1016/j.procs.2020.02.171. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920302945

[9] M. Brooks, S. Basu, C. Jacobs, and L. Vanderwende, "Divide and correct: using clusters to grade short answers at scale," in *Proceedings of the first ACM conference on Learning @ scale conference*, ser. L@S '14. New York, NY, USA: Association for Computing Machinery, Mar. 2014. doi: 10.1145/2556325.2566243. ISBN 978-1-4503-2669-8 pp. 89–98. [Online]. Available: https://doi.org/10.1145/2556325.2566243

[10] E. H. Mory, "Feedback research revisited," in *Handbook of research on educational communications and technology, 2nd ed.* Mahwah, NJ, US: Lawrence Erlbaum Associates Publishers, 2004, pp. 745–783. ISBN 978-0-8058-4145-9

[11] B. S. BLOOM, "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring," *Educational Researcher*, vol. 13, no. 6, pp. 4–16, Jun. 1984. doi: 10.3102/0013189X013006004 Publisher: American Educational Research Association. [Online]. Available: https://doi.org/10.3102/0013189X013006004

[12] Z. S. Harris, "Distributional Structure," *WORD*, vol. 10, no. 2-3, pp. 146–162, Aug. 1954. doi: 10.1080/00437956.1954.11659520. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/00437956.1954.11659520

[13] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, Jan. 1988. doi: 10.1016/0306-4573(88)90021-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0306457388900210

[14] R. Hecht-nielsen, "III.3 - Theory of the Backpropagation Neural Network**Based on "nonindent" by Robert Hecht-Nielsen, which appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE." in *Neural Networks for Perception*, H. Wechsler, Ed. Academic Press, Jan. 1992, pp. 65–93. ISBN 978-0-12-741252-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780127412528500108

[15] S. Ruder, "An overview of gradient descent optimization algorithms," arXiv, Tech. Rep. arXiv:1609.04747, Jun. 2017, arXiv:1609.04747 [cs] type: article. [Online]. Available: http://arxiv.org/abs/1609.04747

[16] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv:1301.3781 [cs]*, Sep. 2013, arXiv: 1301.3781. [Online]. Available: http://arxiv.org/abs/1301.3781

[17] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," *arXiv:1607.01759 [cs]*, Aug. 2016, arXiv: 1607.01759. [Online]. Available: http://arxiv.org/abs/1607.01759

[18] L. M. L. Cam and J. Neyman, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability: Weather modification*. University of California Press, 1967, google-Books-ID: IC4Ku_7dBFUC.

[19] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, Jul. 2002. doi:

10.1109/TPAMI.2002.1017616 Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[20] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transactions on Information Theory*, vol. 21, no. 1, pp. 32–40, Jan. 1975. doi: 10.1109/TIT.1975.1055330 Conference Name: IEEE Transactions on Information Theory.

[21] F. Murtagh and P. Legendre, "Ward's Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward's Criterion?" *Journal of Classification*, vol. 31, no. 3, pp. 274–295, Oct. 2014. doi: 10.1007/s00357-014-9161-z. [Online]. Available: https://doi.org/10.1007/s00357-014-9161-z

[22] T. Kodinariya and P. Makwana, "Review on Determining of Cluster in K-means Clustering," *International Journal of Advance Research in Computer Science and Management Studies*, vol. 1, pp. 90–95, Jan. 2013.

[23] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Information Retrieval*, vol. 12, no. 4, pp. 461–486, Aug. 2009. doi: 10.1007/s10791-008-9066-8. [Online]. Available: https://doi.org/10.1007/s10791-008-9066-8

[24] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3336–3341, Mar. 2009. doi: 10.1016/j.eswa.2008.01.039. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741740800081X

[25] T. Zesch, M. Heilman, and A. Cahill, "Reducing Annotation Efforts in Supervised Short Answer Scoring," in *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*. Denver, Colorado: Association for Computational Linguistics, Jun. 2015. doi: 10.3115/v1/W15-0615 pp. 124–132. [Online]. Available: https://aclanthology.org/W15-0615

[26] M. Wolska, A. Horbach, and A. Palmer, "Computer-Assisted Scoring of Short Responses: The Efficiency of a Clustering-Based Approach in a Real-Life Task," in *Advances in Natural Language Processing*, ser. Lecture Notes in Computer Science, A. Przepiórkowski and

M. Ogrodniczuk, Eds. Cham: Springer International Publishing, 2014. doi: 10.1007/978-3-319-10888-9_31. ISBN 978-3-319-10888-9 pp. 298–310.

[27] L. H. Cross and R. B. Frary, "Hodgepodge Grading: Endorsed by Students and Teachers Alike," *Applied Measurement in Education*, vol. 12, no. 1, pp. 53–72, Jan. 1999. doi: 10.1207/s15324818ame1201_4. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1207/s15324818ame1201_4

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Oct. 2011. [Online]. Available: https://hal.inria.fr/hal-00650905

TRITA-EECS-EX-2022:361