Karlsruhe Institute of Technology

Carnegie Mellon University

Enhancing Lecture Transcript Comprehensibility By Recognising Mathematical Formulae

Bachelor's Thesis of

Fabian Martin

at the Department of Informatics
Institute for Anthropomatics and Robotics

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Prof. Dr. Tamin Asfour

20. Juli 2018 – 19. November 2018

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

This thesis was written during an exchange at Carnegie Mellon University in Pittsburgh (Pennsylvania) and was kindly supported by a scholarship from DAAD.

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karslruhe, 24th of October, 2018

. . . .

(Fabian Martin)

Acknowledgments

First, I would like to thank Prof. Dr. Alexander Waibel for his valuable advice. Also, I'm very grateful for the opportunity to have been a part of the CLICS and InterACT exchange program, which made it possible for me to write my bachelor thesis at Carnegie Mellon University in Pittsburgh. It was a memorable experience that has greatly benefitted me both professionally and personally.

Abstract

This thesis is a contribution to the field of natural language processing, in particular, the subfield that deals with processing mathematical formulae in natural language. The objective of this work is to develop a program that recognises mathematical formulae in lecture transcripts, matches them with their mathematical representation and attaches this representation to the transcript in order to make it more comprehensible. On a test dataset consisting of 148 sentences taken from 4 different lectures by 2 lecturers, the system correctly recognises 91% of mathematical formulae spoken by the lecturer and only incorrectly matches a formula for every three correctly matched formulae.

Zusammenfassung

Diese Bachelorarbeit ist ein Betrag zum Gebiet der Natürlichen Sprachverarbeitung, im besonderen dem Teilgebiet, welches sich damit beschäftigt mathematische Formeln in natürlicher Sprache zu verarbeiten. Das Ziel dieser Arbeit ist es, ein Programm zu entwickeln das mathematische Formeln in Transkripten von Vorlesungen erkennt, ihrer mathematischen Repräsentation zuordnet und diese Repräsentation dem Transkript beifügt, sodass es leichter verständlich wird. Auf einem Testdatensatz der aus 148 Sätzen besteht, die aus 4 Vorlesungen von 2 verschiedenen Lektoren stammen, erkennt das System 91% der vom Lektor gesprochenen mathematischen Formeln korrekt und nur für jede dritte korrekt erkannten Formel, wird eine Formel falsch erkannt.

Contents

1. Introduction **1**

2. Theoretical Background **2**

2.1. Symbol, LaTeX command and Formula . . . 2
2.2. Classification . . . 2
2.2.1. Support Vector Classification (SVC) . . . 3
2.2.2. Decision Tree . . . 3
2.2.3. Random Forest . . . 3
2.2.4. XGBoost . . . 4
2.3. Word Embeddings . . . 4
2.4. Sentence Similarity Measures . . . 4
2.4.1. Levenshtein Distance . . . 4
2.4.2. Hamming Distance . . . 5
2.4.3. Jaro Distance . . . 5
2.4.4. SequenceMatcher . . . 5
2.5. Evaluation Metrics . . . 5
2.5.1. Correct and Incorrect Classification . . . 6
2.5.2. Positive Predictive Value (PPV) . . . 6
2.5.3. Negative Predictive Value (NPV) . . . 6
2.5.4. Recall . . . 6
2.5.5. False Positive Rate . . . 7
2.5.6. ROC AUC . . . 7
2.5.7. Precision-Recall AUC . . . 7

3. Related Work **8**

3.1. Recognition and Translation of spoken formulae . . . 8
3.2. Word Classification . . . 9
3.3. Machine Translation . . . 10
3.4. String Similarity Measures . . . 11

4. Assumptions and Limitations **12**

4.1. Assumptions . . . 12
4.2. Limitations . . . 13

5. Data **14**

5.1. Data for the System as a whole . . . 14
5.2. Class examples . . . 14
5.3. Frequently occurring problems . . . 16

6. Design and Implementation 18
6.1. System Overview . . . 18
6.2. Design Decisions . . . 19
6.2.1. Sub-formulae . . . 19
6.2.2. Pseudo-Natural Language . . . 20
6.3. Subsystems . . . 21
6.3.1. Classification Subsystem . . . 21
6.3.2. Translation Subsystem . . . 25
6.3.3. Matching Subsystem . . . 26

7. Evaluation 30
7.1. Evaluation Metrics . . . 30
7.1.1. Metrics for the System as a whole . . . 30
7.1.2. Metrics for the Classification subsystem . . . 32
7.2. Classification Algorithms . . . 33
7.3. First Approach . . . 36
7.4. Classification Base Rules . . . 37
7.5. Sentence Similarity Measures . . . 38
7.6. Longest Consecutive Matching Word Count (LCMWC) . . . 39
7.7. Penalties . . . 39
7.7.1. Variable-Difference Penalty . . . 39
7.7.2. Gap-Size Penalty . . . 41
7.8. Valid Match Threshold (Similarity Threshold) . . . 44
7.9. Final Results . . . 45

8. Conclusion 48

Bibliography 49

A. Appendix 52
A.1. Data for Evaluation Graphs . . . 52
A.1.1. Classifiers . . . 52
A.1.2. Sentence Similarity Measures . . . 52
A.1.3. Variable-Difference Penalty . . . 52
A.1.4. Gap-Size Penalty . . . 53
A.1.5. Valid Match Threshold . . . 53
A.2. List of Translation Rules with covered symbols . . . 53

1. Introduction

It is clear that speaking comes much more natural to us than writing or entering information with the help of a keyboard. Hence, Automatic Speech Recognition (ASR) has been an important field of study since its emergence in the 1950s. ASR has seen immense progress and enabled many new technologies that used to be considered science-fiction just a few decades ago. For instance, it is now possible to talk to your computer as you would to another human being.

It is not surprising that technologies like Apple's Siri and Amazon's Alexa have become so popular. They enable us to communicate with our computer more easily and are especially useful for people with disabilities that prevent them from entering information by use of a keyboard or touchpad.

A simplified computer input is just one of the benefits of ASR. It has also enabled passive information input. Where usually, information was only transmitted to another person or an audience, ASR can now help to make additional use of this information without requiring the user to change their behaviour. An example of this is the Lecture-Translator, which allows presenters to record their speech while giving a talk. It then creates a transcript of spoken words and translates it into a multitude of different languages. Just by using a microphone and feeding the audio recording into the Lecture-Translator, the speaker can create additional value without any additional effort.

One area that has not seen much progress, however, is the recognition of mathematical formulae in spoken language. While there have been some developments, a satisfying solution has not yet been found, as it is a very difficult problem to find the correct mathematical formula that a human being is talking about. [1]

The objective of this bachelor thesis is to present a first step towards solving the problem of recognising mathematical formulae in natural language. The use case is the application as part of the Lecture-Translator. A system was developed that recognises formulae in the transcript of a lecturer's speech and matches it with the corresponding mathematical formula on the lecture slide. The formulae can then be inserted into the transcript to make it more comprehensible.

Chapter 2 introduces the theoretical background necessary to fully understand this thesis. It is followerd by chapter 3, which presents related work. Chapter 4 deals with the assumptions made during the creation of the system and the limitations of this thesis. Chapter 5 describes the data used to evaluate the system and in chapter 6, the system's design and implementation are discussed. Chapter 7 evaluates the system's performance and gives a detailed account of the results. Chapter 8 concludes the thesis and gives an outlook of possible future work.

2. Theoretical Background

This chapter introduces the basic concepts required to understand this bachelor thesis and define terms that are used throughout this work. The descriptions are concise and not meant to be complete.

2.1. Symbol, LaTeX command and Formula

A **symbol** is the smallest entity in mathematics. It corresponds to a character in a natural language. Examples of symbols are: $+, *, \alpha, x$.

A **LaTeX command** is the LaTeX code for any symbol.
Examples are: +, *\ast*, *\alpha*, *x*.

A **formula** is an entity constructed using the symbols and formation rules of a given language. It expresses information symbolically. For clarity, three different terms are used for formulae, depending on the underlying type of language.

- A **natural formula** is a formula in a natural language.
  Examples are: *X squared*, *The integral from minus infinity to infinity of y dy.*
- A **mathematical formula** is a formula in the language of mathematics.
  Examples are : X^2, $\int_{-\infty}^{\infty} y\ dy$.
- A **LaTeX formula** is the LaTeX code for a mathematical formula.
  Examples are: *X ^ { 2 }*, *\int _ { - \infty } ^ { \infty }* y dy.

2.2. Classification

Classification is the task of identifying which category a new observation belongs to. It is a supervised learning approach that requires correctly labeled training data, from which it generalises and learns how to classify a previously unseen data point. Binary classification is classification with exactly two categories.

Cross-validation is a statistical method of evaluating machine learning algorithms by dividing data into two segments: one used to train a model and the other one used to validate it [2]. The goal is to evaluate the model on data that was not used during training, which can give an indication whether the model suffers from overfitting or other biases [3].

Overfitting is the characteristic of a learning algorithm to correspond too closely to the training dataset and fail to generalise on future observations. Overfitting can be detected when a high accuracy is achieved on the training data but a low accuracy on the validation data [4].

Most common machine learning algorithms have a set of hyperparameters that must be determined before training the algorithm [5]. Hyperparameter optimisation is the problem of choosing the hyperparameters for a given learning algorithm. The choice of hyperparameters can have a significant impact on the algorithm's performance [5]. Grid search is a technique for optimising hyperparameters, where all combinations for a list of parameter values for each relevant parameter are compared and the best result is chosen.

In the following, the classification algorithms compared in this thesis are introduced:

2.2.1. Support Vector Classification (SVC)

Support Vector Machines (SVM) are typically used for learning classification, regression or function ranking [6]. In the case of classification, the term Support Vector Classification (SVC) is often used.

Each data point is represented as an n-dimensional vector. In the case of binary classification, a data point belongs to one of two classes. Binary SVC works by linearly separating the data points using a hyperplane. If there are multiple possibly hyperplanes, the one with the largest margin is chosen. The margin is the summation of the shortest distance from the hyperplane to the nearest data point of both categories [6]. The hyperplane with the largest margin is likely to generalise best.

For non-linear classification problems, the kernel trick is used which maps the input into higher dimensional feature space. In the feature space, the classification problem is linear and can be solved by using hyperplanes [6].

2.2.2. Decision Tree

A Decision Tree Classifier uses a decision tree to predict the class of a data point by observing its properties. The classifier learns simple decision rules inferred from the features of the training dataset and builds a decision tree.

In a decision tree, each inner node splits the instance space into two or more sub-spaces according to the values of a set of attributes [7]. A branch represent a conjunction of features that lead to a leaf node, which represents a class label.

2.2.3. Random Forest

Random Forest is an ensemble learning method for classification, regression and similar tasks. It uses Bootstrap Aggregating (Bagging), which is a technique used to minimise variance and can therefore help avoid overfitting. Bagging works by splitting the training data into multiple parts, training an algorithm on each part and averaging the output of each algorithm.

Random Forest classification works by constructing many decision trees independently using a random sub-sample of the training data. The mode of the results of individual trees is the result of this classifier. The mode, in this context, is the class that appears most often.

It shares many characteristics of Decision Trees Classifiers but can reduce the amount of overfitting by making use of the Law of Large Numbers thanks to the independent randomness used in the construction of the trees [8].

2.2.4. XGBoost

XGBoost is a learning algorithm that uses Gradient Tree Boosting, which is an ensemble learning method used for classification and regression problems. Gradient Tree Boosting is the technique of training multiple decision trees sequentially, such that each new tree reduces the errors made by the previous tree. By combining many classifiers that are only slightly better than random guessing, a classifier can be constructed that achieves very good results [9].

2.3. Word Embeddings

Word embeddings are used to extract semantic features from words and transform them into vectors of real numbers. Words are typically treated as arbitrary encodings that do not provide useful information regarding their relationship to other words. Word embeddings map semantically similar words to nearby points in a vector space.

Word2vec is one of the best performing and most commonly used word embeddings [10]. It uses a two-layer neural network to map words into vector space. This neural network is trained on a large dataset in order to learn the linguistic context of words.

Word2vec can use two different model architectures to produce a vector representation of words: continuous bag-of-words (CBOW) and skip-gram. CBOW predicts words from a window of surrounding context words, while skip-gram works in the opposite way and uses the current work to predict the surrounding context words.

2.4. Sentence Similarity Measures

In this subchapter, the sentence similarity measures compared in this thesis are introduced. All of these metrics are syntactic similarity measures that measure the difference of two strings based on syntax, not on semantics.

2.4.1. Levenshtein Distance

Levenshtein Distance measures the minimum amount of insertions, deletions and substitutions necessary to change one string into the other. The more different two strings are, the higher is the Levenshtein Distance.

For two strings a and b, the Levenshtein Distance is calculated as:

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j)+1 \\ \operatorname{lev}_{a,b}(i,j-1)+1 \\ \operatorname{lev}_{a,b}(i-1,j-1)+1 \end{cases} & \text{otherwise.} \end{cases}$$

where $\operatorname{lev}_{a,b}(i,j)$ is the levenshtein distance between the first i characters of a and the first j characters of b [11].

2.4.2. Hamming Distance

Hamming Distance measures the number of characters that differ between two strings. Normally Hamming Distance is only defined on strings of equal length but in this thesis, a version is being used that counts extra characters as differing. The more different two strings are, the higher is the Hamming Distance.

2.4.3. Jaro Distance

Jaro Distance measures the minimum amount of single-character transpositions required to change one string into the other. Jaro Distance is a value between 0 and 1, where 0 signifies no similarity at all and 1 is an exact match.

The Jaro Distance of two strings a and b is calculated as:

$$d_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

Where:

- m is the number of matching characters and
- t is half the number of transpositions [12].

2.4.4. SequenceMatcher

SequenceMatcher is based on a syntactic similarity measure technique called “gestalt pattern matching” first introduced by Ratcliff and Obershelp. This technique tries to find matches that make intuitive sense to humans by finding the longest contiguous matching substring, using it as an anchor and applying the same idea recursively to the sequences to the left and the right of this substring [13].

2.5. Evaluation Metrics

This subchapter introduces the metrics used to evaluate the developed system.

2.5.1. Correct and Incorrect Classification

A **true positive** is an observation that was correctly classified as positive.

A **true negative** is an observation that was correctly classified as negative.

A **false positive** is an observation that was incorrectly classified as positive.
The underlying data point is actually negative.

A **false negative** is an observation that was incorrectly classified as negative.
The underlying data point is actually positive.

In this thesis, the class of formulae is referred to as the positive class, while the class of non-formulae is referred to as the negative class.

The following confusion matrix illustrates the definitions.

	Predicted: Negative	Predicted: Positive
Actual: Negative	True Negative	False Positive
Actual: Positive	False Negative	True Positive

2.5.2. Positive Predictive Value (PPV)

Positive Predictive Value, also called Precision, measures how many of the observations classified as positive are correct. Mathematically, it is defined as follows:

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

2.5.3. Negative Predictive Value (NPV)

Negative Predictive Value measures how many of the observations classified as negative are correct. Mathematically, it is defined as follows:

$$\frac{\text{True Negatives}}{\text{True Negatives} + \text{False Negatives}}$$

2.5.4. Recall

Recall measures how many of the observations that were positive were actually classified correctly. Mathematically, it is defined as follows:

$$\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

2.5.5. False Positive Rate

False Positive Rate measures how many of the observations that were negative were misclassified as being positive. Mathematically, it is defined as follows:

$$\frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

2.5.6. ROC AUC

ROC AUC is the two-dimensional area underneath the Receiver Operating Characteristic (ROC) curve from (0,0) to (1,1). The ROC curve is a graph that plots the false positive rate on the x-axis and recall on the y-axis. A higher area under the curve signifies a better model.

2.5.7. Precision-Recall AUC

Precision-Recall AUC is the two-dimensional are underneath the Precision-Recall curve from (0,0) to (1,1). The Precision-Recall curve is a graph that plots recall on the x-axis and precision on the y-axis. A higher area under the curve signifies a better model.

3. Related Work

3.1. Recognition and Translation of spoken formulae

There has not been much success so far in recognising and translating spoken formulae into mathematics.

The first paper that dealt with the problem was published by Kevin Lin and R. Fateman in 2004. They built a simple program called SKEME (Symbolic Keyboard and Mouse Editor) that allowed users to enter mathematical equations with keyboard and mouse input. It also supported voice recognition as an additional feature but was mainly focused on non-verbal input methods. They were able to recognise simple formulae like “a plus b” or “script capital A” [1] [14].

Fateman et al. developed a system called ‘Math Speak & Write’ in 2004 that was also able to recognise simple formulae and was more robust than SKEME. Their approach was to use speech and handwriting input for the same formula and combine these two methods to increase recognition accuracy. Their system had similar limitations and was not considered to be more than “demoware” [1] [15].

In 2007, McClellan published a tool called ‘MathTalk’ that was able to recognise mathematical formulae spoken by a user. However, it had some significant limitations: It required pauses before and after each operation, the input language was unnatural (military names like ‘foxtrot’ had to be used for f and most importantly, the user had to pre-train every single symbol before it was usable [1] [15] [16] [17].

Fateman et al. concluded in 2013, that popular speech recognition models like Siri that use keywords or phrases from spoken utterances in the context of a calendar, address book or similar situation, are not suitable for mathematical formula input. They fail to produce the desired results for variations as simple as changed variable names. A more sophisticated model has to be developed that “recognises the peculiar nature of mathematical discourse” [1].

There have been some approaches to translating English into mathematics that showed acceptable results, however, these have dealt with very simple problems and clean input data [18].

Automatic Mathematic Problem Solvers have been developed that can translate clearly stated mathematical problems in a mixture of natural and mathematical language (as in pre-university exam questions) into a form that can then be automatically processed by computers. They show good results. However, similar results cannot be expected on data from the Lecture-Translator

[19].

Matching Math to descriptions oftentimes works by finding special phrases (e.g. *X is Y*, *X is given by Y*) or finding patterns in the natural language and using them to identify places where formulae are mentioned. This presupposes that coherent sentences are available, so that noun phrases or other grammatical structures can be found, that give an indication about the position of formulae. However, this is rarely the case with input data from the Lecture-Translator and consequently it's not an option to use similar approaches in this thesis [20].

Solving the problem of recognising and translating spoken formulae into mathematics is a very difficult problem. However, this thesis' problem does not necessarily require solving this problem. Because the system is used as part of the Lecture-Translator, the formulae that need to be recognised are the formulae on the lecture slides and are therefore known. These formulae can be translated into natural language, which is a much simpler problem than translating natural language into mathematics, and can then be matched with the spoken utterances of the lecturer. This allows finding the mathematical representation for a natural formula without solving the above-mentioned problem.

The problem of recognising mathematical formulae in the transcript of a lecture can therefore be divided into three smaller problems:

1. Finding natural formulae in the transcript (Classification)
2. Translating all possible formulae into the natural language (Machine Translation)
3. Matching the spoken formulae with their corresponding mathematical formulae (Matching)

3.2. Word Classification

Machine learning is commonly used for classification tasks. Many different machine learning models are used in practice and there is no best model for all problems; which approach achieves the best results depends on the nature of the problem [21].

Neural models have shown particularly good results for text classification tasks [22]. However, these models require a large training dataset in order to achieve good results that are statistically significant [23]. Training neural networks on small amounts of data can lead to significant performance losses due to overfitting [21]. Additionally, neural networks require a lot of computation time and infrastructure investments in order to be trained within a reasonable amount of time [24]. Shallow learning techniques can overcome the problems of neural models and perform better on small datasets [21].

When dealing with highly imbalanced training data, classifiers generally perform poorly because they aim to minimise the overall error rate and end up favouring the majority class. This often leads to almost all instances being classified as the majority class [25] [26].

There are two common techniques that deal with training on highly imbalanced data. The first one is cost-sensitive learning, where a higher cost is assigned to the misclassification of the minority class. The second technique is using a sampling technique, where either the majority class is downsampled or the minority class oversampled. Downsampling is a technique that artificially decreases the size of a class, while oversampling increases the size of a class. Both of these approaches can lead to an increase in performance for algorithms trained on highly imbalanced data [26].

Shallow learning algorithms that have shown good results when trained on imbalanced data using one of these two techniques are: Decision Tree [27], Random Forest [26], XGBoost [28] and Support Vector Machine [25].

In order to classify words, a common approach is to convert them into vectors which can then be used as input for machine learning models. Word embeddings can be used to extract semantic features from words and transform words into vectors [29] [30] [31]. One of the best and most commonly used word embeddings is word2vec [10].

3.3. Machine Translation

Two major approaches to machine translation are Rule-Based Machine Translation (RBMT) and Neural Machine Translation (NMT). NMT has shown better results than RBMT in many cases, when translating between two natural languages [32]. However, Neural Machine Translation requires large datasets in order to draw statistical conclusions.

Mathematical language has different characteristics than natural language. Features such as gender, plural and part of speech don't exist for formulae and cannot be used for coreference resolution. This makes it hard to apply standard natural language processing methods to formulae [20]. Additionally, there is not enough labeled data available for the coreference relations between formulae and texts. Therefore, commonly used machine learning techniques cannot be applied to mathematical language without expensive human annotations [20].

For the problem of this thesis, a machine translation system that can correctly translate one language into another is not necessary. Firstly, the input of the system are LaTeX formulae and not complicated natural language sentences, which makes it significantly easier to define rules for translation. Secondly, a coherent output is not necessary. The translated mathematical formula is only used to match the spoken utterances of the lecturer. Therefore, what matters is that the match results are good; not that the translation is a syntactically correct sentence in the natural language. And lastly, good results for the system can already be achieved by translating only the most frequent mathematical symbols, which make up most of the formulae that are likely to be used in a lecture setting. This substantially reduces the required manual labour to build a Rule-Based Machine Translation system. This removes many of the problems of classical RBMT and thus makes it a viable translation system for this thesis.

3.4. String Similarity Measures

The two groups of commonly used approaches to measuring similarities between strings are semantic similarity measures and syntactic similarity measures.

Semantic similarity measures match semantically related words. Traditionally, semantic similarities are calculated with the help of manually compiled dictionaries such as WordNet. However, a lot of terms are not covered by these dictionaries which makes this approach unusable for cases in which many of these terms are used [33]. Mathematics is one of these cases.

Oftentimes semantic similarity is calculated by using the percentage of co-occurrence of two terms or by looking at the context in which two words are used [33]. This can cause problems in this thesis' application. In mathematics, the contexts of the words "minus" and "plus" are unlikely to be different from one another. In natural language, a word with a negative connotation and a word with a positive connotation are oftentimes surrounded by other negative/positive words. This makes it possible to learn the sentiment of these words and therefore their semantic relationship to other words. In mathematics, this is harder because the same formula can easily be used with a plus and a minus sign. Thus, it is hard to learn the semantics of a mathematical symbol by looking at the context in which a symbol is used.

"Minus" and "plus" could be considered related words by a system that looks at co-occurrence, because they tend to be used in similar situations. However, the string similarity measure should calculate a big difference between these two words, otherwise the wrong formula can very easily be chosen, especially because it is not a rare case in which the same formula with a minus and with a plus sign are both present on the same lecture slide.

Therefore syntactic similarity measures are the focus of this work. Even though they also present their issues, they are less likely to match completely different formulae and tend to give results that are close to the correct formula.

Some of the most commonly used syntactic similarity measures are: Levenshtein Distance, Hamming Distance and Jaro Distance. Additionally, the python library "difflib" has implemented a similarity measure called SequenceMatcher that has become popular for the comparison of complicated strings.

4. Assumptions and Limitations

4.1. Assumptions

The objective of the system is to supplement a lecture transcript so that it is more easily understandable when dealing with mathematical formulae. The system finds natural formulae in a lecture transcript and matches the corresponding mathematical formula from the lecture slides. The results are used as part of the Lecture-Translator to optionally show the user the mathematical formula in addition to the words that describe the said formula. The integration of this system into the Lecture-Translator website was not part of this thesis. However, it could be implemented as an optional feature that highlights the area of text that describes a formula and then shows the corresponding mathematical formula when hovering over it with the mouse.

Because this thesis' goal is merely to improve comprehensibility, formula recognition does not have to be perfect. If the recognised formula has an additional symbol i.e. '=' but is otherwise correct, it is still going to help the user understand the content of the transcript. The same is true for the location of the formula; if an additional word of the transcript is highlighted as part of the formula, the benefit to the user is not substantially decreased. The evaluation metrics take this into consideration. These metrics are explained in more detail in Chapter 7.1.

The system assumes two kinds of input. The first input are the sentences uttered by the lecturer. The sentences could, in theory, be from any text, but in this thesis, they are sentences from a transcript of a lecture that covers topics related to mathematics. The test data is taken from the transcript from the Lecture-Translator website[1] and separated into sentences by splitting it at every dot ('.'). The Lecture-Translator occasionally mistakes a sentence that contains a long pause for multiple sentences. This means that a formula might be spread over two or more sentences. This makes it difficult to detect the formula when only handling one sentence at a time. However, because the system detects sub formulae instead of complete formulae (explained in chapter 6.2.1), the system can detect the part of the formula spoken in each sentence and give a sufficiently good result to the user in most cases.

The second input is a list of all LaTeX formulae that the lecturer could possibly be talking about in the sentence. These are all formulae from the lecture slides shown during the time of utterance of each sentence. The Lecture-Translator has a hyperlinked transcript of the lecture and the corresponding video, which allows the user to click on a word and see the lecture from the point in time, at which this word is spoken. This makes it very easy to find the slide that was shown during the utterance of a sentence. There is a plethora of tools that convert images

[1]https://lecture-translator.kit.edu/

of formulae into their LaTeX equivalent. An example is Mathpix[2], which allows users to take a screenshot of a mathematical formula and convert it into LaTeX code. Mathpix also has an API that allows automated formula conversion.

Another assumption is that the program is not used as part of a realtime application and the running time therefore is not crucial. The problem of finding all sub formulae in a given sentence, as well as in the corresponding lecture slide, and then correctly matching them is by nature exponential in the size of the sentence, as well as the formulae on the slide. Therefore the program can take a long time for long sentences or slides that contain many formulae. It is not the focus of this thesis to develop a program that can deal with big input sizes in a short amount of time. However, by restricting the input to single sentences and assuming formulae from only one slide at a time, the running time is kept within reasonable bounds.

4.2. Limitations

The system is limited to German and English and cannot currently process input in any other language. The focus of this thesis is on German because the data from the Lecture-Translator is significantly cleaner in the original language of the lecture, which was German in the case of the test data. Because a rule-base translation system is chosen for translating latex-formulae into spoken language, adding support for additional languages requires a non-trivial amount of work. For languages that are similar to German and English, the translation rules do not have to be altered, however a lot of language specific strings have to be changed. Adding a language like Chinese would require a considerable amount of work. It is also the case that, unlike a neural approach, a rule-based approach does not handle previously unseen symbols well. Translation rules for the most common symbols were created and the program can therefore correctly translate formulae in most cases, however there will definitely be formulae containing infrequently occurring symbols, that will not be translated sufficiently well. It therefore requires continuous updating of the translation algorithm, in order to achieve a very high accuracy for a large set of data. A rule-based approach was chosen mainly because the required amount of data and time to develop a neural translation approach was not available within the scope of this bachelor thesis.

What exactly constitutes an improvement in the comprehensibility of the lecture transcript was determined by the author and not tested experimentally. Also, which formula a lecturer actually talks about in the test data for the system was determined by the author by watching the lectures.

[2]https://mathpix.com/

5. Data

5.1. Data for the System as a whole

The entire data for the system as a whole consists of 196 sentences that is taken from 5 lectures by 2 courses from 2 different lecturers. The courses are computer science courses that have a significant amount of mathematical content. Both courses are German and therefore all test data is in German. The test data is only taken from 4 of these lectures because one of them is used for training. This reduces the amount of sentences that can be used for testing to 162.

The data taken from the Lecture-Translator is rather unclean. It is often the case that words are incorrectly transcribed and sometimes sentences are completely incomprehensible even for a human reader. Therefore, the data is classified into three classes: *unclean*, *somewhat clean* and *clean*. Sentences in the unclean class contain multiple errors that make them difficult to understand even for a human being[1]. The somewhat clean sentences contain some errors that make it moderately difficult to understand the content but generally allow a human reader to make sense of it. The clean data barely contains any errors and is easy to understand for a human.

Out of these 162 sentences, 148 sentences are classified as being somewhat clean or clean. The system is not expected to be able to correctly recognise unclean sentences. All other sentences should ideally be recognised. Therefore, the test data contains 148 sentences. 109 out of these sentences do not contain any formulae and the remaining 39 sentences contain a total of 43 formulae. Some sentences contain multiple formulae. The data was collected by hand from the Lecture-Translator website.

5.2. Class examples

1. **Clean sentences**

   I
   - Sentence in German:
     Auf die rechte Seite hier verschoben worden ist das rücken wir aus durch dieses minus X null, das ist die Funktion eben nach rechts verschoben.
   - Sentence in English:
     Here on the right side has been postponed is that we back from through this minus X 0, that is the function just shifted to the right.

[1]based on the experience of the author and two volunteers

- Formula in Sentence:
  $-x_0$

II
- Sentence in German:
  Und das ist definiert als das Integral von minus unendlich bis plus unendlich von F von T minus Tau mal G von Tau mal Lterter, also kontinuierliche Variante.
- Sentence in English:
  And this is defined as the completely of minus infinitely to infinitely plus of F of T minus Tau times G of the Tau times Lterter, continuous variant.
- Formula in Sentence:
  $\int_{-\infty}^{\infty} f(t-\tau)g(\tau)$

III
- Sentence in German:
  Hier wir geben dem ganzen einen sigmoid Funktion das heißt wir haben eins durch eins plus E hoch minus X J die Sie zwar nicht.
- Sentence in English:
  Here we give the entire a Sigmoid function, that is, we have one divided by one plus E to the power of minus X J, which you may not.
- Formula in Sentence:
  $\frac{1}{1+e^{-x_j}}$

2. **Somewhat clean sentences**

I
- Sentence in German:
  Das heißt wir haben von minus unendlich bis plus unendlich berechnen wir die Fläche unter dieser Funktion, das ist die Dirac Funktion über.
- Sentence in English:
  That is, we have plus infinitely of minus infinitely until we calculate the surface, this is the Dirac function about under this function.
- Formula in Sentence:
  $\int_{-\infty}^{+\infty}$

II
- Sentence in German:
  Das ist meine Ausgabewert und wenn ich die nach D Y J ableite dann bekomme ich wieder sehen die zwei kürzt.
- Sentence in English:
  This is my output value, and if I derive the to D Y J, then I get again see the two abbreviate.
- Formula in Sentence:
  $\frac{\partial E}{\partial y_j}$

III
- Sentence in German:
  Wenn ich wieder von R0, Plus, Nach, N0, plus Abbildung dann, ist, das, ganze so.
- Sentence in English:
  If I again of R0, plus, after, N0, plus mapping, then, is, the whole thing.

- Formula in Sentence:
  $\mathbb{R}_0^+ \rightarrow \mathbb{R}_0^+$

3. **Unclean sentences**

I
- Sentence in German:
  Abgeleitet das haben wir hier gerade bestimmt die nach Y D Y J haben wir schon das ist das hier mal D Y J nach D.
- Sentence in English:
  Derived this here, we have just determines the D Y J with respect to Y D Y J, we have already, this is the here after D.
- Formulae in Sentence:
  i. ∂y_j
  ii. $\cdot\frac{\partial y_j}{\partial x_j}$

II
- Sentence in German:
  X J Y nach Y J Mal D X J, das heißt wir brauchen die Ableitung von der sigmoid Funktion und die haben wir hier müssen also Ableiten, D Y J nach D X J Und wie machen wir das auch eine ganze Folie dafür.
- Sentence in English:
  X J, Y to Y J times D X J, that is, we need the derivative of the Sigmoid function, and here we have, therefore, have to derive D Y J with respect to D X J, and how we do this also a whole slide for this.
- Formulae in Sentence:
  i. $\frac{\partial y_j}{\partial x_j}$
  ii. $\frac{\partial y_j}{\partial x_j}$

III
- Sentence in German:
  Oder X ist nicht Element von von Jahr, dann streichen wir diese stilisierte Ii einfach durch.
- Sentence in English:
  Or X is not element of of year, then we delete this stylized ii simply through.
- Formula in Sentence:
  $x \notin A$

5.3. Frequently occurring problems

In the following, some frequent problems that occur in the Lecture-Translator data are listed, that make it difficult to recognise the correct formula.

- Variable names are messed up (e.g. ’E’ becomes ‘I’, ’N’ becomes ‘R’)
- Variables are confused with regular words (e.g. ’A’ becomes ‘Year’)

- The lecturer repeats certain parts of a formula. This can result in natural formulae like

  X J, Y to Y J times D X J

  that represents the formula
  $$\frac{\partial y_j}{\partial x_j}$$
  This is clearly hard to recognise.

- Lecturer skips important words (e.g. 'integral' in the first example of the class *somewhat clean*)

6. Design and Implementation

This chapter presents the design and implementation of the system. First, a general overview of the system is provided and the fundamental design decisions are discussed. Second, the subsystems are explained in detail and information is given about their implementation and optimisation.

6.1. System Overview

Input
As described in Chapter 4, the input of the system is a sentence from the transcript of a lecture in a natural language, as well as a list of all LaTeX formulae on the lecture slide that were shown during the utterance of the sentence.

Step 1: Classification
Firstly, the system uses a binary classification algorithm in order to classify the words in the sentence into one of the following two classes:

1. formula word
2. non-formula word

The formula words are then grouped together to create a sentence that describes one particular formula. This formula sentence is further processed by the matching algorithm.

Step 2: Translation
Secondly, the LaTeX formulae are translated into the natural language by a rule-based machine translation algorithm. Natural languages are highly ambiguous and multiple ways to describe a mathematical formula are possible. Therefore, the algorithm creates multiple translations that contain the most frequent pronunciations of each symbol in the formula. The list of all possible translations for every formula is passed on to the matching algorithm.

Step 3: Match
Lastly, the formula sentence is compared with all possible translations of every formula using a sentence similarity measure. The LaTeX formula with the best matching translation is then chosen as the corresponding formula, if it exceeds a minimum similarity threshold; otherwise the formula sentence does not match and will be discarded.

Output
If there is a match, the formula sentence, its location in the sentence and the LaTeX code for the corresponding formula are returned as the output of the program.

6.2. Design Decisions

There are two fundamental design decisions that were made for the development of this system. This subchapter explains these decisions, their reasons and the consequences that they entail.

6.2.1. Sub-formulae

In most cases, the lecturer does not read aloud an entire formula from the slide. Showing the entire formula under those circumstances can be confusing to the reader, especially when the formula is large. Because the goal of this system is to improve the transcript's comprehensibility, it is designed to only match the part of the formula that is actually being mentioned instead of the entire formula.

For example, if the lecture slide contains the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

but the lecturer only mentions the radicand

b squared minus four a c

, the system finds the sub-formula of the quadratic formula and matches

$$b^2 - 4ac$$

. The following implementation details are chosen to realise this design decision.

First, instead of combining all formula words into a single formula sentence, the system creates multiple formula sentences. This is required because the lecturer might be talking about multiple formulae in any given sentence. This is implemented by combining all consecutive formula words into a formula sentence. If there are multiple groups of consecutive formula words in a sentence, the system will therefore try to match multiple formulae for this sentence.

For example, the following is a representation of a sentence, where 'F' represents a formula word and 'N' represents a non-formula word:

$$N\ N\ F_1\ F_2\ F_3\ N\ N\ F_4\ N\ F_5\ F_6\ N$$

The consecutive F's are then combined to three formula groups:

- Group 1: F_1, F_2, F_3
- Group 2: F_4
- Group 3: F_5, F_6

The matching algorithm will try to match each formula group to an individual mathematical formula. However, formula groups with less than or equal to two formula words are discarded. In this example, Group 2 and 3 will not be considered in the matching algorithm. Formula descriptions in a natural language that are up to two words in length can easily be understood without the corresponding mathematical formula. The decision to require a minimum length of formula groups significantly reduces the amount of false positives without impacting the transcript's comprehensibility.

Additionally, Formula groups will be split on commas because the Lecture-Translator inserts commas when the speaker takes a break in between words[1]. This is usually a sign that the subsequent formula words are part of a different formula. However, even if the subsequent words are part of the same formula, the correct formula can still be matched because the matching algorithm tries all possible combinations of formula groups.

A formula sentence is a combination of all formula groups of a sentence that allows merging of multiple formula groups into one. For the previous example, this means that all possible combinations are:

- Group 1: F_1, F_2, F_3, Group 2: F_4 and Group 3: F_5, F_6
- Group 1: F_1, F_2, F_3, F_4 and Group 2: F_5, F_6
- Group 1: F_1, F_2, F_3 and Group 2: F_4, F_5, F_6
- Group 1: $F_1, F_2, F_3, F_4, F_5, F_6$

There are a total of 2^{n-1} possible formula sentences for n formula groups.

The second change that is necessary is to split each LaTeX formula into all valid sub-formulae. If the LaTeX formula is considered to be a sentence, this splitting is done by finding all possible subsentences and checking if the resulting mathematical formula is valid. What exactly constitutes a valid mathematical formula and how this is implemented is described in more detail in Chapter 6.3.3.1. All sub-formulae are then translated into the natural language and the sub-formula with the highest similarity to the formula sentence will be chosen as the best matching formula.

6.2.2. Pseudo-Natural Language

Translating a LaTeX formula into a natural language is a difficult task due to the ambiguity of natural languages. However, the difficulty can be significantly reduced by loosening the requirements of the translation result. Because the translated formula is only used to match the formula sentence found in the transcript sentence, it is not necessary to create a perfectly coherent sentence.

[1]only if the sentence contains a maximum of 5 commas. The Lecture-Translator sometimes messes up and inserts commas after every word. These sentences are excluded.

Therefore, the system translates the LaTeX formulae into a pseudo-natural language that is not syntactically correct but contains all semantically important information. The classification algorithm also focuses only on words that have semantic value and aims to classify filler words as non-formula words.

Because natural formulae are matched with mathematical formulae by using a syntactic similarity measure, removing any semantically unimportant words decreases the average formula distance and therefore increases the likelihood of finding a match.

6.3. Subsystems

6.3.1. Classification Subsystem

The classification subsystem takes a sequence of words in natural language as an input, preprocesses each word in the sentence and transforms it into a vector representation. Optionally, these words can be checked for certain characteristics by a set of Base Rules and directly classified if they fall into certain groups. This removes the possibility of classification errors. The vector representation is fed into a trained classification algorithm that classifies each word as a formula or non-formula word. The formula words of a sentence are grouped together into formula sentences, which are returned by the classification subsystem. The following subchapters explain these steps in detail.

6.3.1.1. Training and Test Data

The binary classification algorithms used in this experiment were trained using supervised learning. This requires labeled training data. The first part of the data is taken from a computer science lecture. Every word in the Lecture-Translator transcript for this lecture was manually labeled. This resulted in 4.321 labeled words, of which 90% were labeled as non-formula words and the remaining 10% as formula words. This data is highly imbalanced, which can be problematic for training classification algorithms. When trained on this data, algorithms tend to develop a strong bias towards classifying words as the majority class, which in this case is the class of non-formula words. To mitigate this effect, class weights are set during the training phase to assign a higher cost to the misclassification of the minority class during training.

Besides the manually labeled lecture data, additional data was added to the dataset that was labeled automatically. This step was taken in order to increase the amount of training data.

The first set of supplementary data consists of the translations for all mathematical symbols that are covered under the rules of the rule-based translation subsystem. As is covered in depth in Chapter 6.3.2, the machine translation algorithm, that converts LaTeX formulae into natural language, uses rules to translate the most frequent mathematical symbols. It therefore has a list of the most common pronunciation for many mathematical symbols. Every word of this list was labeled as a formula word and added to the dataset. This resulted in 287 additional formula words.

The second set of additional data are the words from 2000 randomly chosen sentences from German news articles from the year 2011. This data was taken from GermanWordEmbeddings[2]. These news article sentences contain a large amount of commonly used words of the German language. These words were all labeled as non-formula words and added to the dataset. There might be some misclassification because the data was not cleaned and might contain some mathematical words. However, because the formula data was oversampled, the effect of these misclassifications are minimal. In this way, 16.489 additional non-formula words were added to the dataset.

Oversampling was used on all formula words in order to improve the training dataset's balance. The data was split in 80% training data and 20% test data before oversampling was used on the training data. The test data was not oversampled. This enables the test data to be used for cross-validation.

After oversampling, the training data contains 22.017 words of which 25.93% are formula words, while the test data contains 4.219 words of which 3.29% are formula words. The distribution in actual lectures is somewhere between 0% to 10% formula words, depending on the lecture.

6.3.1.2. Classification Algorithms

There are many machine learning algorithms that can be used for binary classification tasks. Neural Networks have been shown to achieve very good results for similar tasks but require a significant amount of training data in order to achieve them. However, not enough labeled data was available and there was not enough time, in the scope of this bachelor thesis, to label the data oneself. Therefore, this thesis uses shallow learning, which requires significantly less data in order to produce good results.

In this thesis, multiple shallow learning algorithms will be compared. As a result of the imbalanced dataset, only algorithms that perform well on imbalanced datasets were chosen. The following algorithms all share this characteristic: SVC, Decision Tree, Random Forest, XGBoost. Except for XGBoost, all classifiers are available in the python library 'sklearn'. XGBoost is implemented in the 'xgboost' library.

These algorithms are analysed in Chapter 7.2 in their default form and with balanced class weights and tuned hyperparameters. The hyperparameters are tuned by using the scikit-learn implementation of Gridsearch and maximising ROC AUC. Gridsearch evaluates the performance of the algorithms with different hyperparameters and chooses the parameter combination with the highest ROC AUC. After comparing all algorithms, the best one is chosen for the task of classifying words into one of the following classes: *formula word* or *non-formula word*.

[2]https://devmount.github.io/GermanWordEmbeddings/

6.3.1.3. Pre-processing words

Instead of feeding the words directly into the classification algorithm, they are first pre-processed by word2vec. The implementation used in this program is by GermanWordEmbeddings[3], which is a word2vec model that uses the skip-gram model architecture. It was pre-trained on over 650 million German words and can recognise more than 600 thousand words [34] . This word2vec implementation creates a vector of 300 features for every word, that can then be used as an input for the classification algorithm. The vector contains semantic information, so that similar words are mapped to similar points in the vector space. Consequently, training the classification algorithm to correctly classify a word increases the probability of correctly classifying semantically similar words as well.

The words are cleaned before being passed to the word2vec algorithm, so that as many words as possible are recognised. Words that are not recognised by word2vec will be directly considered non-formula words instead of being classified by the classification algorithm. There are three steps to cleaning a word. Firstly, the words a stripped of punctuation. Secondly, the special characters ‘ä’, ‘ö’, ‘ü’ and ‘ß’ are converted to a form consisting of only the 26 standard characters of the alphabet (e.g. ’ä’ -> ‘ae’, ‘ß’ -> ‘ss’). And lastly, the most common math symbols are transformed into their natural language equivalent (e.g. ‘+’ -> ’plus’). This is necessary because the Lecture-Translator sometimes inserts these mathematical symbols into the transcript instead of using natural language.

After cleaning and pre-processing all the words of a sentence, the list of vectors representing this sentence is then processed by the classifier.

6.3.1.4. Base rules

There are words that should always be considered formula words and equally there are words that should never be considered formula words. Classifiers can make mistakes and misclassify these words. Therefore, creating a set of base rules that check if a word has certain characteristics, that put them into one of these groups, before feeding it into the classifier can improve classification results.

Additionally, there are many words that can be used in a mathematical context, as well as in a non-mathematical context. When training the classifier on data from lectures, these words tend to be used more frequently in a non-mathematical context and are therefore biased towards being classified as non-formulae. This is especially severe if the classifier is also trained on additional data from news articles that is entirely classified as non-formula data, as is done in this thesis. However, by observing of the context for these types of words, the likelihood of classifying them correctly can be increased. In the following, two groups for these types of words are presented.

The first group are *context-dependent formula words*. These are words that can be used in a mathematical, as well as a non-mathematical context. Examples of these words are: “plus”,

[3]https://devmount.github.io/GermanWordEmbeddings/

"equal", "times". Words in this group should only be classified as formula words if they are used in a mathematical context. A word is considered as being used in a mathematical context if it is either classified as a formula word by the classifier or in vicinity of a formula word. In the first case, the classification algorithm has most likely seen cases of similar words being used in a mathematical context, which increases the likelihood that this word is being used in the same manner. In the second case, the word is very likely part of the description of the mathematical formula. Being in vicinity of a formula means that there is a formula word within distance of two words in either direction.

The second group are *context-required formula words.* These are words that are very frequently used, that also have a mathematical meaning. Examples of these words are: "of", "from", "to". Words in this groups should only be classified as formulae if they are in vicinity of a formula. They are used to connect formula words that would otherwise be separated by a gap.

This thesis will experiment with the following base rules, that can be grouped into four groups. The first group a word matches is the group it belongs to.

Group 1 — Formula Word Rules:
If a word has one of the following characteristics, it is classified as a formula word:

- It consists of only one character
- It is either a number word ('one', 'two', ..) or an ordinal number word ('first', 'second', ..)
- It is a letter of the Greek alphabet ('alpha', 'beta', ..)
- It is a summation word ('sum', 'integral', ..)
- It is a trigonometric or otherwise special function ('sine', 'arccosine', ..)

Group 2 — Non-Formula Word Rules:
If a word is one of the 1000 most frequently used words in the language and not commonly used in a mathematical context, it is classified as a non-formula word. Whether a word is commonly used in a mathematical context was determined by the author.

Group 3 — Context-Dependent Formula Word Rules:
If a word has one of the following characteristics, it is classified as a context-dependent formula word.

- It is a negation ('not', 'no')
- It is a mathematical conjunction ('from', 'over', ..)
- It is an accent that is often used in mathematics ('hat', 'tilde', ..)
- It is used for mathematical operators ('plus', 'equals', ..)

Group 4 — Context-Required Formula Word Rules:
If a word is one of the 1000 most frequently used words that is also commonly used in a mathematical context, it is classified as a context-required formula word.

A version of the classification subsystem that applies these rules will be compared to a version that only uses the classifier in chapter 7.4. The same chapter also demonstrates the effect of using additional data from news articles on the classification results.

6.3.2. Translation Subsystem

The input to the translation subsystem is a LaTeX formula. Multiple translation rules are applied to this formula until finally all frequently used LaTeX symbols are converted into the natural language. Each rule translates a certain type of symbol and passes the resulting partly translated formula to the next rule. Because natural languages are ambiguous, there are multiple possible pronunciations for many mathematical symbols. Therefore, some rules generate multiple translations. Hence, after applying all rules, many different translations for the original formula are created. Because there is not a rule for every possible LaTeX symbol, some infrequently used symbols might not be translated during this process. These symbols are then stripped of all special characters and kept as part of the translation.

Following is a list of all groups of symbols that are translated by the system:

- Summation (Sum, Integral, Product, Co-Product)
- Function
- Root
- Power
- Fraction
- Operators
- Greek symbols
- Trigonometry and special functions
- Accents
- Numbers
- Variables
- Spacing
- Braces

A list of the symbols that belong to each group can be found in Chapter A.2

Example for the translation result of the LaTeX formula $2^n + f(x)$:

- in German: 2 hoch n plus f von x
  in English: 2 to the power of n plus f of x
- in German: zwei hoch n plus f von x
  in English: two to the power of n plus f of x
- in German: 2 n plus f von x
  in English: 2 n plus f of x
- in German: zwei n plus f von x
  in English: two n plus f of x
- in German: 2 hoch n plus f x
  in English: 2 to the power of n plus f x
- in German: zwei hoch n plus f x
  in English: two to the power of n plus f x
- in German: 2 n plus f x
  in English: 2 n plus f x
- in German: zwei n plus f x
  in English: two n plus f x

As the example demonstrates, the focus is not on creating perfectly coherent formulae in natural language but on translating the semantically important information.

The resulting list of possible translations are returned by the translation subsystem.

6.3.3. Matching Subsystem

The matching subsystem takes a sentence from the transcript and all LaTeX formulae shown during the time of the sentence's utterance as an input. The formulae are split into all valid sub-formulae, which are then translated by the translation subsystem. The sentences, after being cleaned, are given as an input to the classification subsystem, which returns the corresponding formula sentences. Thereafter, the matching subsystem compares each formula sentence with every possible translation of every LaTeX sub-formula and finds the best match by using a sentence similarity measure. This similarity measure can be very simple or consist of complex logic that takes matching subsentences and penalties into account. The result of this subsystem are the best matching mathematical formulae for all natural formulae recognised in the transcript sentence. All these steps are explained in detail in this chapter.

6.3.3.1. Preparation

The first step of the matching subsystem is to split every LaTeX formula into all possible valid sub-formulae, as introduced in Chapter 6.2.1. A LaTeX sub-formula is valid, if all of the following characteristics hold true:

1. The formula is a valid parenthesis expression. That means that for every opening parenthesis, there is a corresponding closing parenthesis of the correct type in the correct order.
   For example, this is a valid parenthesis expression:

   (a [b + c] - d)

   , while this isn't:

) [a + b) - c]

2. The formula contains at least three semantically informative LaTeX words. Parentheses and special LaTeX formatting symbols are not considered to hold semantic information on their own.

3. The formula does not start with a binary operator (e.g. '=' or '*\in*)' or end in LaTeX words that require further context (e.g. '*\frac*', '*\int*').

Here is an example for all possible valid sub-formulae for the LaTeX formula *a (b + c) = 12 ^ 3*:

- LaTeX formula: *a (b + c)*
  Mathematic formula: $a(b+c)$
- LaTeX formula: *a (b + c) = 12*
  Mathematic formula: $a(b+c)=12$
- LaTeX formula: *a (b + c) = 12 ^ 3*
  Mathematic formula: $a(b+c)=12^3$
- LaTeX formula: *(b + c)*
  Mathematic formula: $(b+c)$
- LaTeX formula: *(b + c) = 12*
  Mathematic formula: $(b+c)=12$
- LaTeX formula: *(b + c) = 12 ^ 3*
  Mathematic formula: $(b+c)=12^3$
- LaTeX formula: *b + c*
  Mathematic formula: $b+c$
- LaTeX formula: *12 ^ 3*
  Mathematic formula: 12^3

All of the valid sub-formulae are then converted into multiple possible translations in natural language by the translation subsystem.

The next step is to clean the sentence. First, punctuation is stripped from the sentence. Afterwards, words written in all uppercase characters are separated into single characters. This is necessary, because the Lecture-Translator transcribes variable names as uppercase characters and sometimes makes the mistake of merging multiple variables together into one. Separating them into each individual character allows the classification subsystem to recognise the formulae and classify them correctly. The sentence is then passed to the classification subsystem, which returns the corresponding formula sentences.

6.3.3.2. Matching Formulae

To find the best match, all formula sentences are considered. For each sentence, the match score of all formula groups is calculated and the combination with the highest average of formula group match scores is chosen. The match score is a natural formula similarity measure that will be explained in more detail in the following subchapter 6.3.3.3.

The match score for each formula group is calculated by comparing it with every possible translation of all valid LaTeX sub-formulae for every formula on the slide. The sub-formula with the best match score is then taken as the corresponding mathematical formula, as long as it exceeds a minimum threshold. This threshold is used to discard formula groups that were either misclassified and don't represent any mathematical formula or represent a mathematical formula that is not on the slides and should therefore not be considered. This value of this threshold has a significant impact on the amount of false positives and false negatives. If the value is set correctly, many misclassification errors can be fixed. This thesis experiments with different threshold values in Chapter 7.8. Only the formula groups that exceed the threshold are considered for the calculation of the average match score.

This approach finds the combination of formula groups that has the highest likelihood of finding all matches. If there are multiple such combinations, one of them is chosen at random.

6.3.3.3. Match Score

This thesis experiments with multiple approaches to calculating the match score. The basis for this score is a syntactic similarity measure that compares a translated LaTeX formula from the slide with a natural formula from the input sentence. Similarity measures that will be compared in Chapter 7.5 are the following: Levenshtein Distance, Hamming Distance, Jaro Distance and SequenceMatcher. The first three measures are string similarity measures that use simple approaches to calculate the distance between two strings. The latter, on the other hand, uses a more sophisticated approach.

For the first two similarity measures, the match score increases for an increased difference in strings, while for the latter two, the match score represents a percentage of similarity, where an increase in difference leads to a decrease in the match score. Because chapter 7.5 will show that the latter two measures achieve better results, it is assumed from here on out, that the match score will be a number between 0 and 1 with 1 being representative of a perfect match.

Additional ideas are considered, that use other characteristics than comparing characters. These allow the match score to be specialised towards solving this thesis' specific problem.

The first supplementary idea is to find the Longest Consecutive Matching Word Count. This is the highest amount of consecutive matching words between the two naturl formulae. Only formula sentences with the highest Count are considered for possible matches. This means that natural formulae that share multiple words in a row are preferred. An example for this approach is the following:

- Formula group:
  X is divided by Y
- Translated LaTeX sub-formula 1:
  X squared divided by Y
  Levenshtein Distance: 7
- Translated LaTeX sub-formula 2:
  X is raised by Y
  Levenshtein Distance: 4
- Naive solution (using Levenshtien Distance):
  Formula 2
- Solution with Longest Consecutive Matching Word Count:
  Formula 1
- More likely the correct solution:
  Formula 1

Another approach is to add penalties that reduce the match score if certain conditions are met. The first penalty is the variable-difference penalty. This penalty subtracts a fixed value from the match score for every variable that is contained in one natural formula but not in the other. A variable is defined as any single-letter word. The second penalty is the gap-size penalty that first calculates the contained gap size of a formula sentence. As explained in Chapter 6.2.1, formula sentences can consist of multiple formula groups merged together. The gap sizes between these initial formula groups are summed up and if the sum exceeds a certain threshold, a penalty will be applied to the difference of the gap size and the threshold. Hence, formula groups that span long parts of the sentence are disadvantaged, which could improve results because these formula groups are likely to contain multiple formulae or unnecessary words that should not be included in the match.

Chapter 7 compares the naive match score approaches with match scores that make use of the supplementary ideas. The best approach is then chosen to find matches.

7. Evaluation

7.1. Evaluation Metrics

This chapter presents the metrics used to evaluate the performance of the system as a whole, as well as, metrics that are used to compare different approaches to the model's design.

7.1.1. Metrics for the System as a whole

The quality of the system is based on its underlying purpose, which is to improve the comprehensibility of the transcript in the Lecture-Translator. The system achieves this by finding natural formulae in the transcript and attaching their mathematical representation. How good the system works is therefore measured by metrics that reflect how well it can solve this task. The system is evaluated on the level of a sentence instead of the level of a word because it is not important that every word is correctly identified as being part of a formula. Rather, it is important that each natural formula in a given sentence is recognised and correctly matched to its equivalent mathematical formula.

Whether the predicted mathematical formula is correct is evaluated using the following metrics. The first two metrics are Correctly Matched Count and Correctly Located Count. For both of these metrics, the count should ideally be as high as the amount of actually existing formulae. Additionally, the system should minimise the amount of incorrect matches. This is being measured using the metric Incorrectly Matched Count. In the following, each of these metrics is described in more detail.

7.1.1.1. Correctly Matched Count

Correctly Matched Count (CMC) is the amount of correctly matched formulae in a given sentence. A formula is correctly matched if the mathematical formula that is predicted to be described by the natural formula is sufficiently similar to the mathematical formula, that the lecturer actually talks about.

Being *sufficiently similar* is defined as fulfilling one of the following two characteristics:

- The LaTeX code of the predicted formula and the LaTeX code of the correct formula have a Levenshtein Distance of less than 5 (not counting spaces).
- The LaTeX code of the predicted formula and the LaTeX code of the correct formula have at most one LaTeX command not in common.

Examples:

- The following two formulae are sufficiently similar based on the first characteristic:

  \frac { x } { y } = 3

  \frac { x } { y }

- The following two formulae are sufficiently similar based on the second characteristic:

  \bigtriangledown f = grad f

  f = grad f

7.1.1.2. Correctly Located Count

Correctly Located Count (CLC) is the amount of correctly matched formulae that are also correctly located in a given sentence. A formula is correctly located if it is a correct match and the predicted location of the natural formula is within a distance of three words from the actual location of the natural formula in the sentence.

Example:

- If the natural formula

  integral of x

  was correctly matched in the sentence

  The integral of x can easily be calculated by hand.

  , it is correctly located if its location is predicted to be in the index interval [0,6] which represents the substring

  The integral of x can easily be

  . Therefore, if the predicted location of the formula

  integral of x

  is the interval of [4,7], it is not considered to be correctly located.

7.1.1.3. Tolerant Correctness

CMC and CLC are both defined, so that small errors are ignored. As explained in Chapter 4, a formula that is slightly wrong can still increase the user's comprehensibility when reading the supplemented transcript. The same is true for a small deviation in the formula's location. The exact values have been chosen by the author after manually looking at hundreds of sentences from lecture transcripts and comparing how different degrees of errors would impact their ability to understand these sentences. A formula that is correctly matched and located, when attached to a lecture transcript, is considered to be beneficial to the reader's comprehensibility.

7.1.1.4. Incorrectly Matched Count

Incorrectly Matched Count is the amount of incorrectly matched formulae in a given sentence. A formula is incorrectly matched if the algorithm detects a natural formula for a sub-sentence, that either is not actually a natural formula or is a different natural formula than the one that was predicted. In the second case, the detected formula was not sufficiently similar to the actual formula.

7.1.2. Metrics for the Classification subsystem

7.1.2.1. ROC AUC

Research has shown that classifiers trained on imbalanced datasets show good results when maximising ROC AUC or Precision-Recall AUC [35] [36]. Which metric is better depends on the importance of certain kinds of errors for the application. The difference is that ROC AUC measures Recall and False Positive Rate, while Precision-Recall AUC measures Recall and Precision.

For highly imbalanced datasets, an increase in the amount of false positives results in a greater decrease for Precision-Recall AUC than for ROC AUC [35]. This is often cited as a reason that Precision-Recall AUC is the better metric for imbalanced datasets [35]. However, in the context of this thesis, this is not the case, because it is significantly more important to maximise the amount of true positives than it is to minimise the amount of false positives, when it comes to the classification algorithm. The reason for this is two-fold:

1. During matching, there are measures taken to discard false positives before they appear in the result. One of these is the valid match threshold described in chapter 7.8 that requires all matches to have a sufficient match score in order to be considered valid. Another measure is to require a formula group to have a minimum length, as described in chapter 6.2.1. This helps to discard wrongly matched formulae that are very short.

2. Because the system is used to supplement the transcript instead of modifying it, an increase in the amount of false positives does not significantly decrease the transcript's comprehensibility. Therefore, false negatives do not have an impact as negative as in many other binary classification tasks.

The following example demonstrates the difference between ROC AUC and Precision-Recall AUC for the classification in the system. The data was taken from an evaluation of the binary classifier RandomForest's performance on the training dataset with different specifications.

RandomForest - Default:
ROC AUC: 0.897
P-R AUC: 0.790
Confusion-Matrix:

	Predicted: Non-Formula	Predicted: Formula
Actual: Non-Formula	15789	518
Actual: Formula	990	4720

RandomForest - Tuned (for ROC AUC) and Balanced:
ROC AUC: 0.922
P-R AUC: 0.728
Confusion-Matrix:

	Predicted: Non-Formula	Predicted: Formula
Actual: Non-Formula	14478	1829
Actual: Formula	250	5460

The default algorithm has significantly less false positives (518 instead of 1829) but also considerably less true positives (4720 instead of 5460) than the tuned and balanced algorithm. The default algorithm has a higher Precision-Recall AUC than the tuned and balanced one, which is reversed for ROC AUC. For the purpose of this thesis, the tuned and balanced algorithm is better, which is why the classifiers are trained to maximise ROC AUC and the algorithm with the highest ROC AUC is chosen. The classifiers' python libraries contain training methods that support maximising this metric during training.

7.1.2.2. Positive and Negative Predictive Value

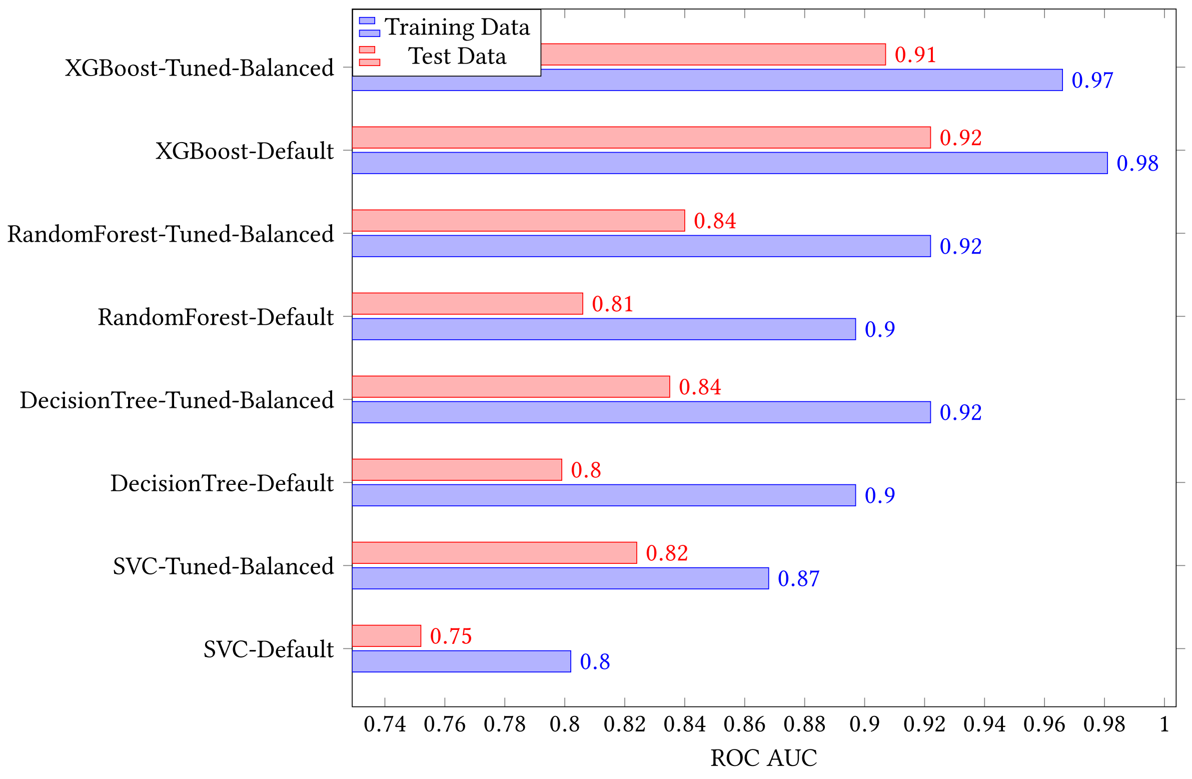
Useful measures in addition to ROC AUC are the Positive Predictive Value (PPV) and Negative Predictive Value (NPV). PPV measures how many of the words classified as formula words are correct and NPV measures how many of the words classified as non-formula words are correct. Because it is so important for this thesis to correctly classify formulae, PPV plays a big role when comparing classification algorithms. If a classifier has the highest ROC AUC, it is not necessarily implied that PPV is also the highest. Therefore, multiple metrics are considered when comparing classifiers. NPV plays a less important role than PPV but can still help to evaluate the performance of an algorithm.

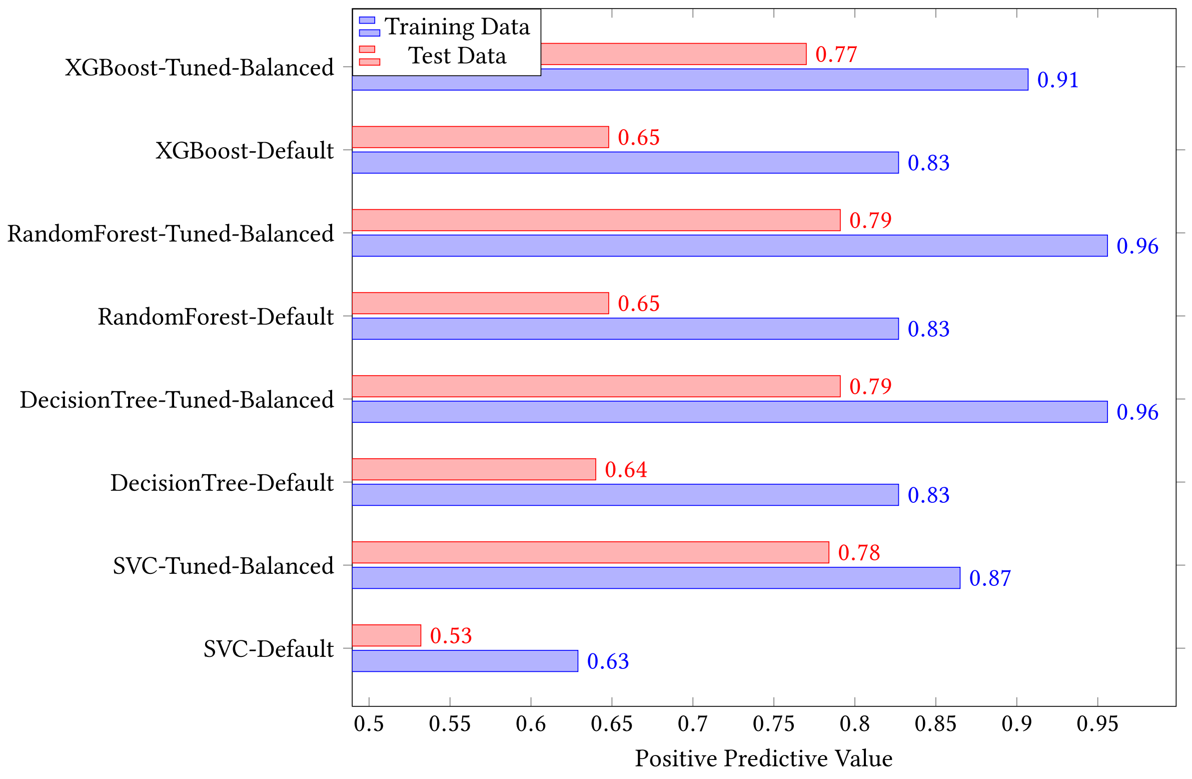
7.2. Classification Algorithms

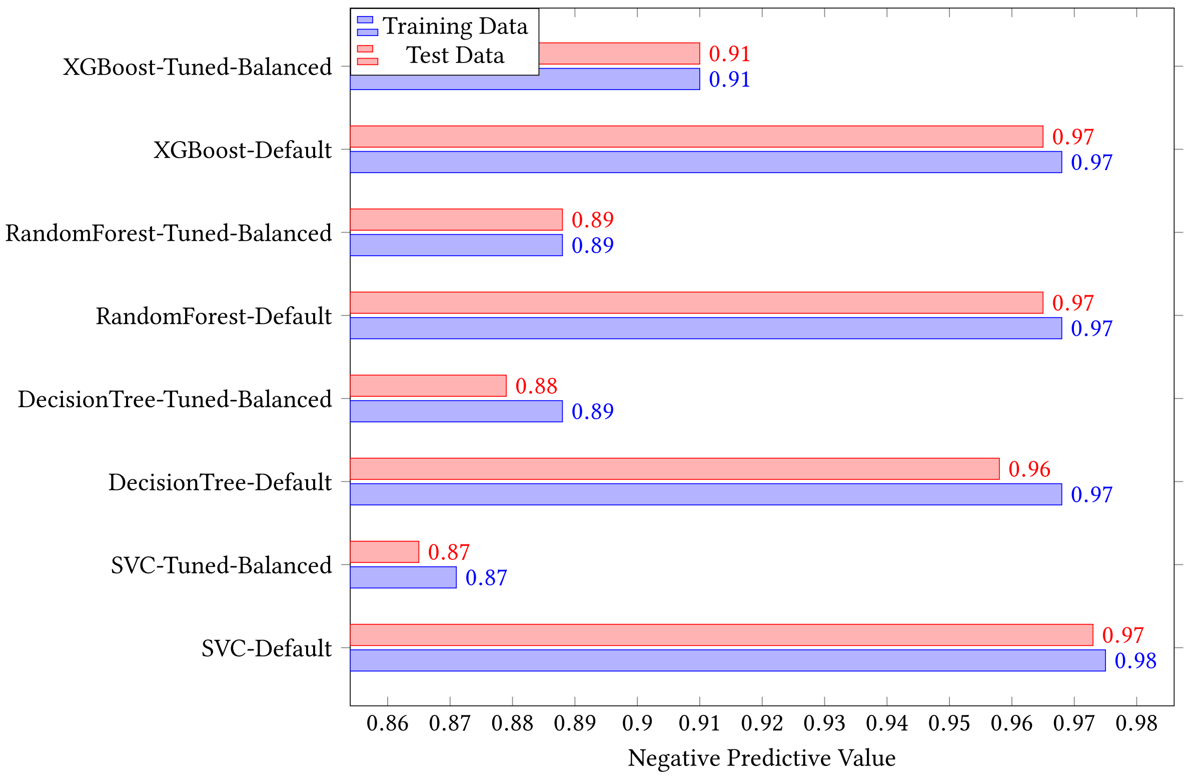
In the scope of this thesis, four binary classification algorithms are compared: SVC, DecisionTree, RandomForest, XGBoost.
The classifiers were trained on the training data described in chapter 6.3.1.1, so that they maximise ROC AUC. All of the algorithms were trained in their default form, as well as tuned and balanced. For hyperparameter optimisation, scikit-learn's GridSearchCV was used to compare commonly used parameter values and choose the best performing specification. The following algorithms are compared in this chapter:

- SVC Default
- SVC Tuned & Balanced
  (Parameters: class_weight=“balanced”, tol=0.001, kernel="rbf")
- DecisionTree Default
- DecisionTree Tuned & Balanced
  (Parameters: random_state=random_seed, class_weight="balanced", criterion="gini", max_depth=None, max_features=5, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=
- RandomForest Default
- RandomForest Tuned & Balanced
  (Parameters: n_jobs=2, random_state=random_seed, class_weight="balanced", criterion="entropy", bootstrap=True, max_depth=None, max_features=9, min_samples_leaf=5, min_samples_split=7)
- XGBoost Default
- XGBoost Tuned & Balanced
  (Parameters: eta: 0.01, max_depth: 17, subsample: 0.9, colsample_by_tree: 0.7)

In the following, the evaluation results of the classifier’s performance on the training data and the test data are presented. Both of these datasets are explained in more detail in chapter 6.3.1.1. It is important to note that the algorithm’s performance on the training data is very important. Many of the most frequent mathematical symbols are in the training dataset and most formulae consist largely of these symbols. This implies that if the classifier can achieve good results on the training data, it will be able to correctly classify most formulae. The test data is important for formulae that consists of multiple symbols that are not frequently used and therefore not part of the training data. A classifier that achieves good results on the training data but only mediocre results on the test data is still going to perform reasonably well most of the time.


Training Data
Test Data
XGBoost-Tuned-Balanced
0.91
0.97
XGBoost-Default
0.92
0.98
RandomForest-Tuned-Balanced
0.84
0.92
RandomForest-Default
0.81
0.9
DecisionTree-Tuned-Balanced
0.84
0.92
DecisionTree-Default
0.8
0.9
SVC-Tuned-Balanced
0.82
0.87
SVC-Default
0.75
0.8
0.74 0.76 0.78 0.8 0.82 0.84 0.86 0.88 0.9 0.92 0.94 0.96 0.98 1
ROC AUC


Training Data
Test Data
XGBoost-Tuned-Balanced
0.77
0.91
XGBoost-Default
0.65
0.83
RandomForest-Tuned-Balanced
0.79
0.96
RandomForest-Default
0.65
0.83
DecisionTree-Tuned-Balanced
0.79
0.96
DecisionTree-Default
0.64
0.83
SVC-Tuned-Balanced
0.78
0.87
SVC-Default
0.53
0.63
0.5 0.55 0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95
Positive Predictive Value



The data shows, that XGBoost (in both specifications) has the best ROC AUC for Training and Test data alike. However, the PPV on the training dataset is considerably worse than for Random Forest and Decision Tree. Because it is very important for the application of this thesis, to have a very high PPV, XGBoost is not considered to be a good classifier. Random Forest and Decision Tree show very similar results in their Tuned and Balanced specification. They have the highest PPV among all classifiers on both the training data and the test data. Additionally, they achieve good results for all other metrics. Random Forest Tuned and Balanced is slightly better than Decision Tree Tuned and Balanced and therefore chosen as the classifier for this thesis.

7.3. First Approach

First, a naive approach is presented, that is used as a benchmark to compare ideas for improvements to. This approach only consists of a classifier, sentence similarity measure, and a valid match threshold. The exact specification is as follows:

- Classifier: RandomForest Tuned and Balanced (see chapter 7.2)
- Sentence Similarity Measure: SequenceMatcher
- No Base Rules
- No Longest Consecutive Matching Word Count
- No Penalties

- Valid Match Threshold = 0.6

The underlying dataset contains 43 actual formulae.

System Specification	Correct Match Count	Correctly Located Count	Incorrect Match Count
First Approach	0	0	3

The results clearly shows that the naive approach is very bad. A system using this approach in a practical application could not expect any improvements in comprehensibility.

7.4. Classification Base Rules

The first addition to the naive system that is being analysed is the use of Base Rules for the classification subsystem, that define rules for the classification of words with certain characteristics. In the following, the naive system is being used as a base system to compare the classification with and without Base Rules.

The following example demonstrates the effect of using these base rules on the classification results. Bold words are classified as formula-words.

1. Sentence:
   in German: *Integral von i ist gleich null bis unendlich von f von x d x*
   in English: *Integral of i is equal to zero to infinity of f of x d x*

2. Classification when trained without news articles:
   in German: ***Integral von i*** *ist* ***gleich*** *null* ***bis unendlich von*** *f* ***von*** *x d x*
   in English: ***Integral of i*** *is* ***equal*** *to zero* ***to infinity of*** *f* ***of*** *x d x*

3. Classification when trained with news articles:
   in German: ***Integral*** *von i ist* ***gleich*** *null bis* ***unendlich*** *von f von x d x*
   in English: ***Integral*** *of i is* ***equal*** *to zero to* ***infinity*** *of f of x d x*

4. Classification when trained with news articles and using Base Rules:
   in German: ***Integral von i ist gleich null bis unendlich von f von x d x***
   in English: ***Integral of i is equal to zero to infinity of f of x d x***

It is clear from this example, that some of the most important formula words are missing when not using Base Rules. This makes it very hard for the system to correctly match formulae. This effect is especially severe when news articles are added to the training data as examples for non-formulae, because then even less words are classified as formula words. The Base Rules classify these important formula words separately and lead to a significant impact in the system's result.

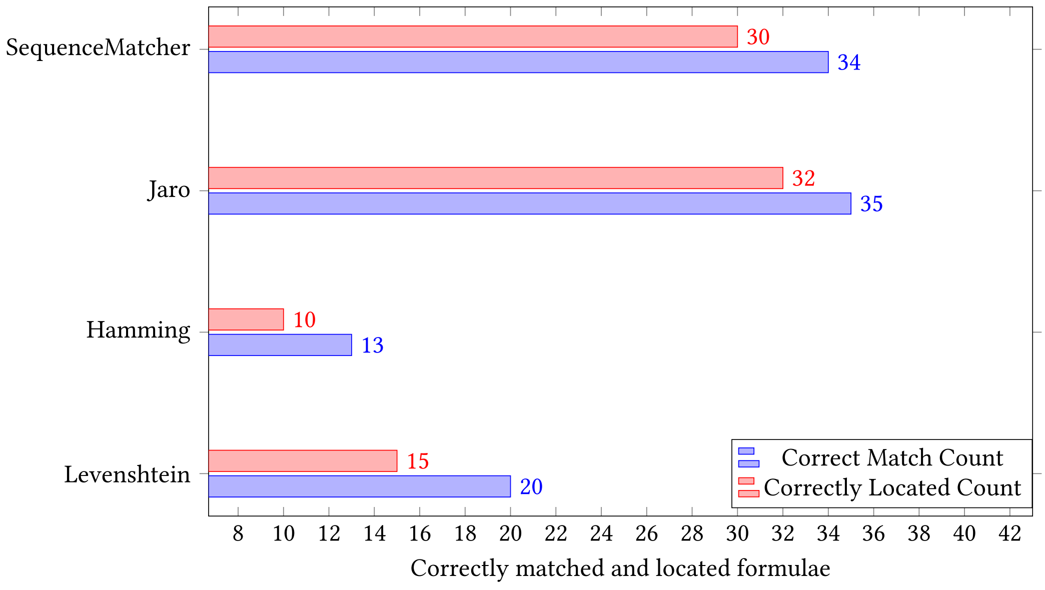
The following observation shows the impact of using Base Rules on the system as a whole. The underlying dataset contains 43 actual formulae.

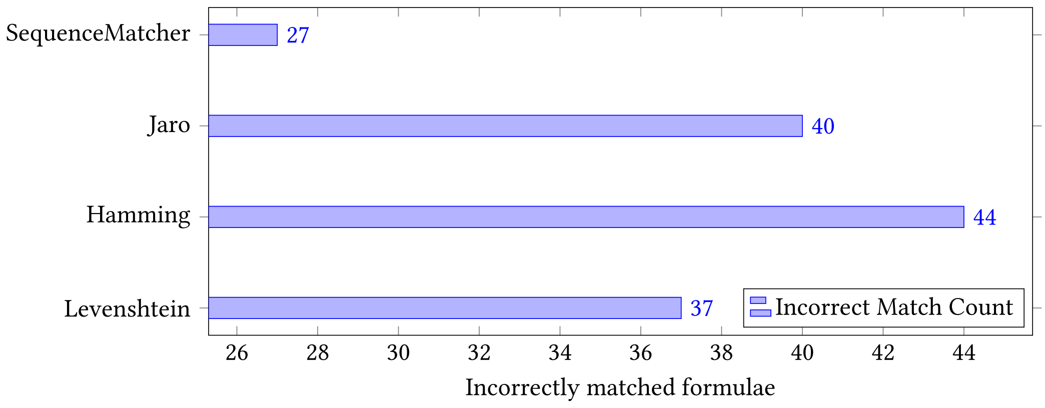
System Specification	Correct Match Count	Correctly Located Count	Incorrect Match Count
Naive System	3	1	3
Naive System with news articles	0	0	3
Naive System with news articles and Base Rules	34	30	27

All three metrics are increased considerably by including Base Rules in the classification subsystem. Therefore, the Base Rules are used in the system developed in this thesis.

7.5. Sentence Similarity Measures

In this subchapter, the different sentence similarity measures presented in chapter 6.3.3.3 are compared. The base system for this comparison is the naive system with Base Rules from the previous subchapter. The underlying dataset contains 43 actual formulae.





The data shows that Levenshtein Distance and Hamming Distance achieve relatively bad results with only 20 and 13 correct matches and 37 and 44 incorrect matches, respectively. When using Hamming Distance, the system matches over 3 formulae incorrectly for every correct match. This makes the similarity measure unfit to be used in an actual application. Levenshtein Distance does not show significantly better results and should therefore also be avoided.

Jaro Distance and SequenceMatcher show much better results. Both of these similarity measures correctly match around 80% of the existing formulae, of which almost all are correctly located as well. The biggest difference between these measures lies in the amount of incorrect matches. SequenceMatcher results in about 32% less incorrect matches than Jaro Distance, while only being slightly worse in the other two metrics, which is the reason why it is chosen as the string similarity measure to be used in the system.

An assumption as to why the first two similarity measures are significantly worse than the latter two is that the first two measures use very simple approaches that might perform well for word comparisons but do not preform well when used on sub-sentences.

7.6. Longest Consecutive Matching Word Count (LCMWC)

This subchapter examines if using Longest Consecutive Matching Word Count improves the system's results. The base system for this comparison is the system using SequenceMatcher from the previous subchapter. The underlying dataset contains 43 actual formulae.

System Specification	Correct Match Count	Correctly Located Count	Incorrect Match Count
Without LCMWC	34	30	27
With LCMWC	37	33	25

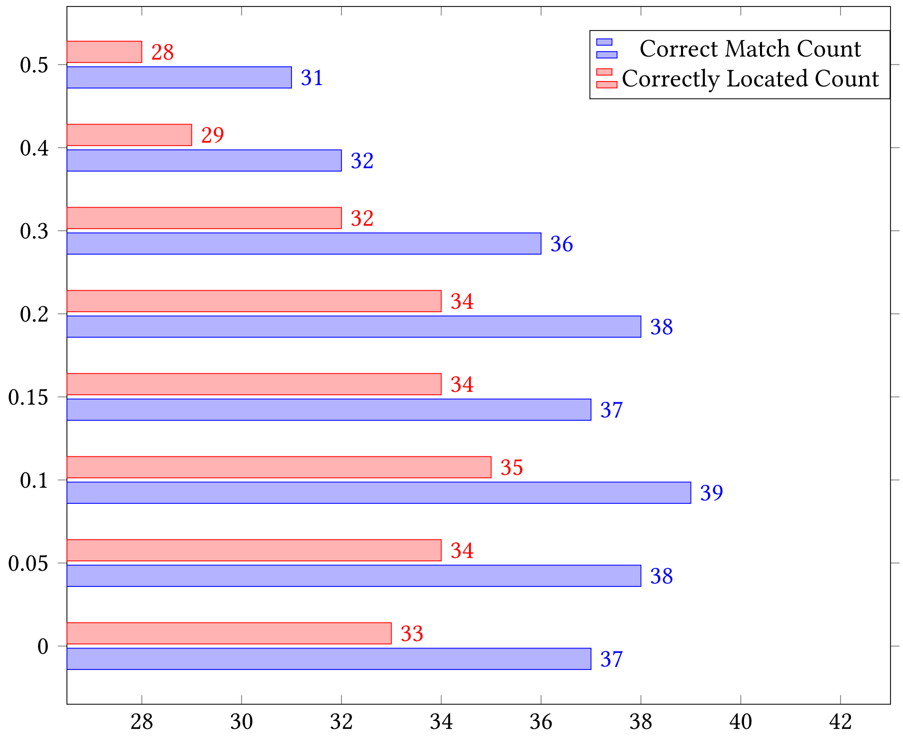
The results show an improvement in all three metrics when Longest Consecutive Matching Word Count is being used to reduce the amount of possible formulae for a match by removing all formulae with a short LCMWC. 3 more formulae are correctly matched and located and 2 fewer formulae are incorrectly matched in the test dataset.

The example in Chapter 6.3.3.3 demonstrated, that there are cases in which using LCMWC can improve the match results. The data from our test set demonstrates that cases like this actually occur in the transcript of lectures. Based on these examinations, Longest Consecutive Matching Word Count is included in the system.

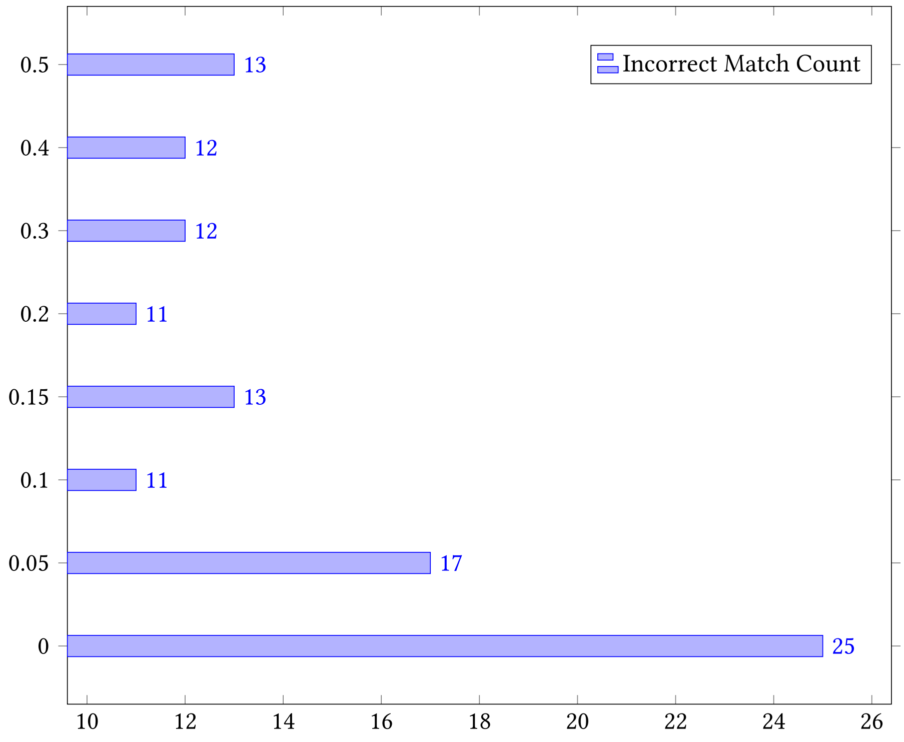
7.7. Penalties

7.7.1. Variable-Difference Penalty

The Variable Difference Penalty reduces the match score by a certain amount for every variable contained in one formula, that is not contained in the other formula. The base system for this comparison is the system using Longest Consecutive Matching Word Count from the previous subchapter. The underlying dataset contains 43 actual formulae.



Correctly matched and located formulae for different Variable-Difference Penalty values



Incorrectly matched formulae for different Variable-Difference Penalty values

The data shows an improvement for all values until 0.2 of Variable-Difference Penalty compared to the approach with no penalty. The biggest improvement is in the decrease of incorrect matches of up to 56%. The amount of correct matches and correctly located matches is also improved slightly for almost all values until 0.2. The penalty values greater than 0.2 show a decrease in correct matches, as well as in correctly located matches. Values higher than 0.5 are unlikely to achieve better results, as implied by the falling trend of correct matches and correctly located matches for an increase in penalty value.

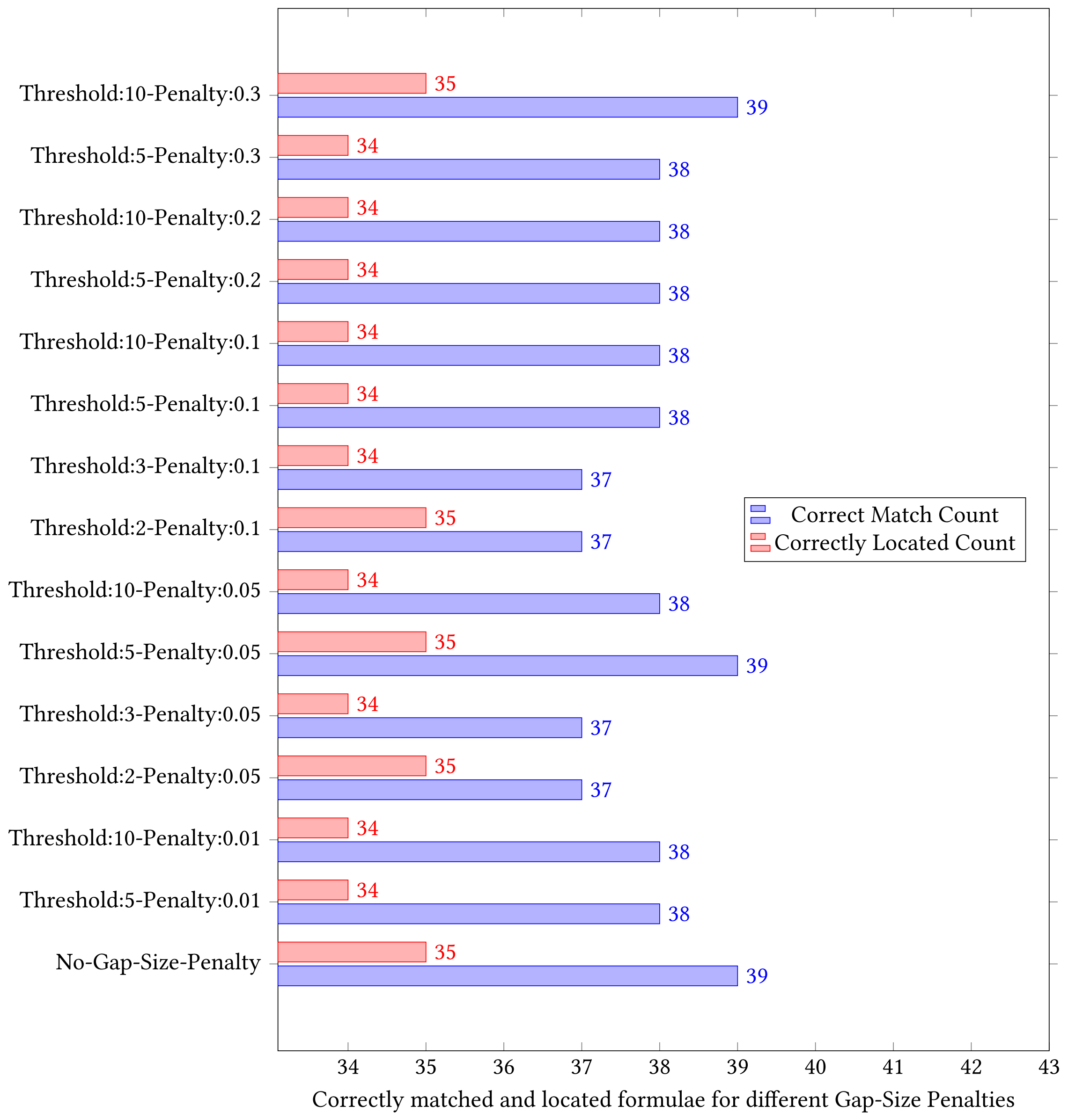
If two natural formulae contain different variables, the likelihood that they both describe the same formula is rather unlikely. This is especially true in a lecture setting, where the lecturer talks about a specific formula that is shown on a lecture slide and is therefore prone to use the same variable names that are used in the formula on the slide. Thus, penalising the use of wrong variables is likely to result in an decrease of incorrect matches. However, if the penalty is too big, more matches fall below the valid match threshold and are therefore discarded. This results in a decrease in correct matches for high penalty values.

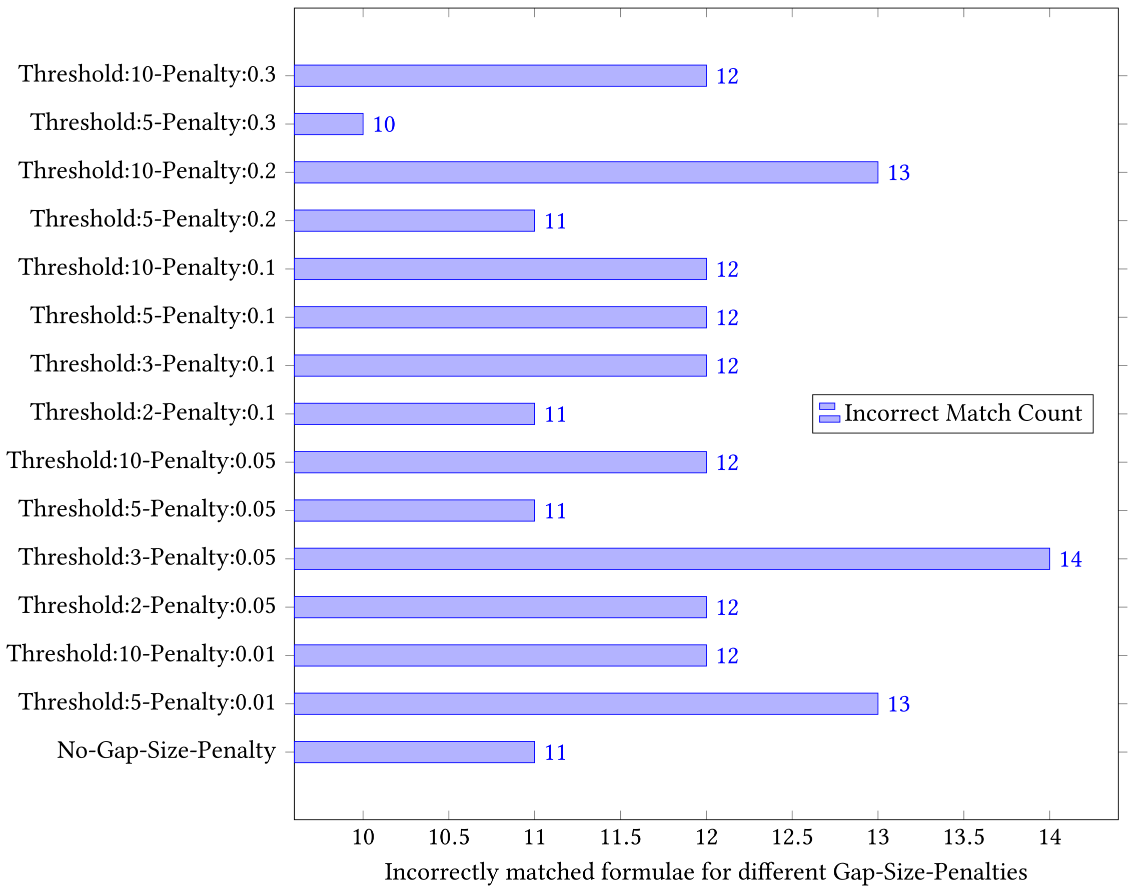
The best results are achieved when using 0.1 as the penalty value, which shows improvements in all three metrics. This value is therefore used for the system.

7.7.2. Gap-Size Penalty

Multiple threshold and penalty values are compared for the gap-size penalty. The threshold determines the maximum contained gap size of a formula group, that does not result in a gap-size penalty. For every additional gap size, the penalty value is subtracted from the match score.

The threshold values 5 and 10 are compared together with the following penalty values: 0.01, 0.05 and 0.1, 0.2, 0.3. For the penalty values 0.05 and 0.1, the threshold values 2 and 3 are also considered. These values were chosen after experimenting on a smaller dataset, which achieved the best results for these values. The base system for this comparison is the system using a Variable-Difference Penalty with the threshold 0.1 from the previous subchapter. The underlying dataset contains 43 actual formulae.


Threshold:10-Penalty:0.3
Threshold:5-Penalty:0.3
Threshold:10-Penalty:0.2
Threshold:5-Penalty:0.2
Threshold:10-Penalty:0.1
Threshold:5-Penalty:0.1
Threshold:3-Penalty:0.1
Threshold:2-Penalty:0.1
Threshold:10-Penalty:0.05
Threshold:5-Penalty:0.05
Threshold:3-Penalty:0.05
Threshold:2-Penalty:0.05
Threshold:10-Penalty:0.01
Threshold:5-Penalty:0.01
No-Gap-Size-Penalty
35 39
34 38
34 38
34 38
34 38
34 38
34 37
35 37
34 38
35 39
34 37
35 37
34 38
34 38
35 39
Correct Match Count
Correctly Located Count
34 35 36 37 38 39 40 41 42 43
Correctly matched and located formulae for different Gap-Size Penalties



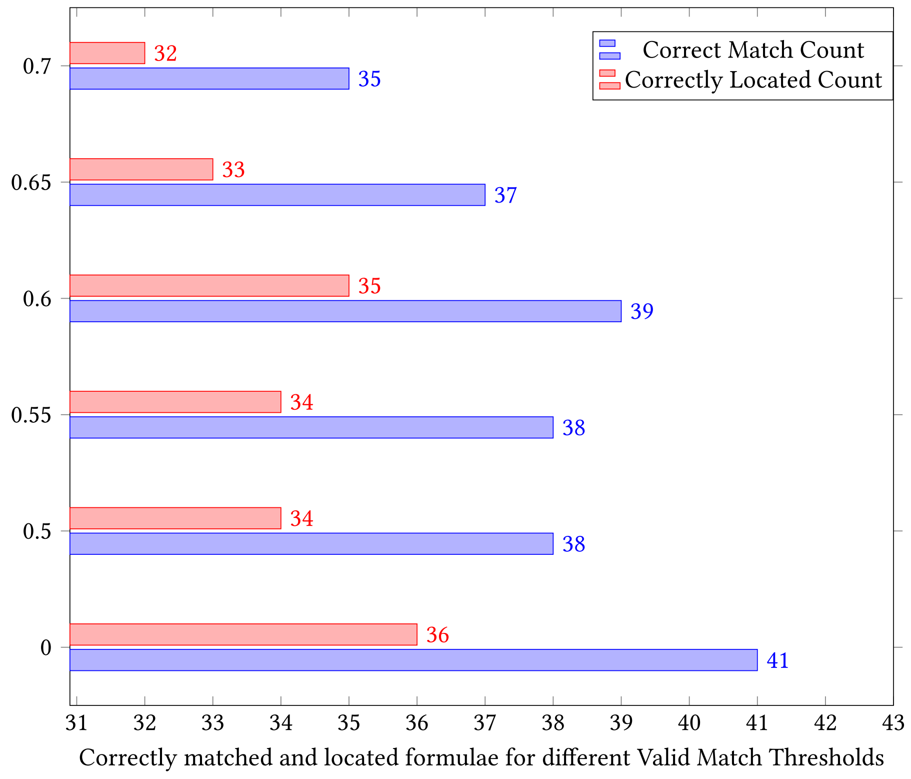
The use of gap-size penalty does not show an improvement in the system's results. Except for the specifications with threshold = 5 and penalty = 0.05, which performs exactly as well as the approach without gap-size penalty, the results have worsened. For almost all of the specifications, the amount of correct matches has decreased, while the amount of incorrect matches has increased slightly. There is no clear trend for the amount of correct and incorrect matches for specifications with different threshold and penalty values, besides the fact that they are all not an improvement to using no gap-size penalty. The same is expected to be true for other threshold and penalty value pairs.

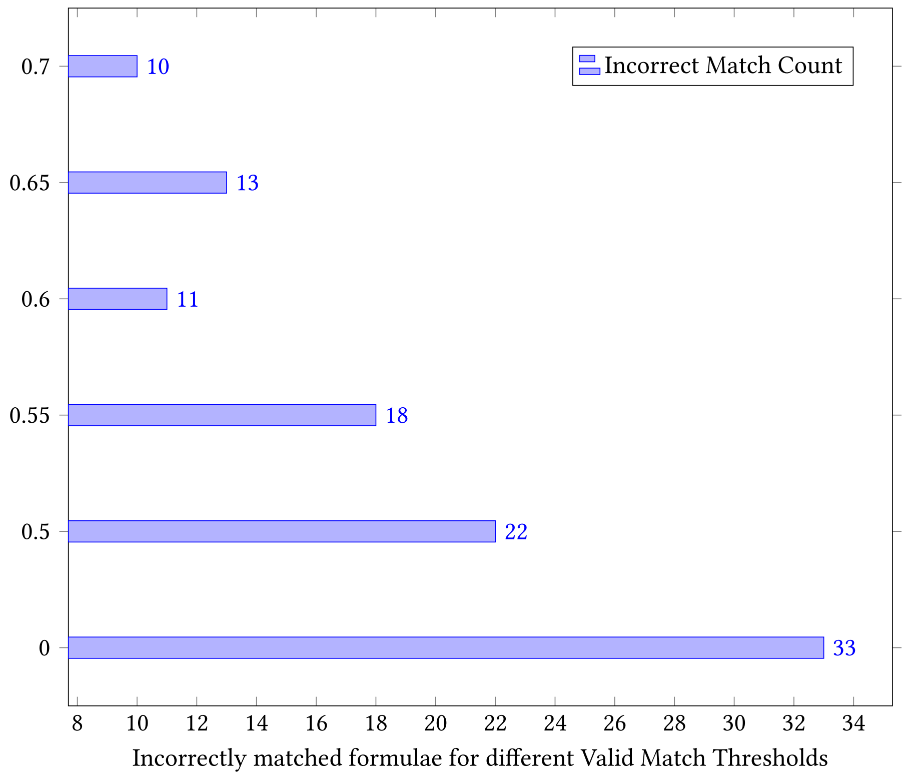
A possible explanation for this observation is that an increase in gap size penalty leads to smaller formula groups and therefore an increase in the amount of formula groups that are so small, that they don't fall within the definition of a formula group and are therefore discarded. In chapter 6.2.1, the requirements of a formula group are discussed in more detail. This effect might offset the improvement that the gap-size penalty brings by discarding overly long formula groups that contain many gaps, which could lead to the results shown above. Additionally, there can be natural formulae that span over long distances, for which the match score is significantly reduced by introducing gap-size penalty. These formulae would have a higher likelihood of being matched correctly without gap-size penalty.

Because the use of gap size penalty has not shown positive results on the test dataset, it will not be used in the system.

7.8. Valid Match Threshold (Similarity Threshold)

The value of the minimum threshold a match score has to exceed so that it does not get discarded is the object of the next analysis. This valid match threshold was introduced in chapter 6.3.3.2. The exact values were chosen after experimenting on a smaller dataset, which achieved the best results for the interval between 0.5 and 0.7. The specification with threshold = 0 represents no Valid Match Threshold. The underlying dataset contains 43 actual formulae.





The data clearly shows that the amount of incorrect matches decreases considerably when using a higher valid match threshold. A decrease of 70% was achieved when using the threshold 0.7. However, the amount of correct matches also decreases slightly.

After a certain point, which the data indicates to be around 0.6, the decrease in incorrect matches does not justify the decrease in correct matches. The amount of incorrect matches decreases by 1 when increasing the threshold from 0.6 to 0.7. Meanwhile, the amount of correct matches decreases by 4.

It is fair to assume that a threshold above 0.7 decreases the amount of correct matches further and leads to worse results than the threshold of 0.6. Furthermore, a threshold below 0.5 has most likely more than 20 incorrect matches and less than 39 correct matches, which makes it worse than the threshold of 0.6. Because of these considerations, the threshold value of 0.6 is chosen for the system.

7.9. Final Results

The final specification of the system is as follows:

- Classifier: RandomForest Tuned and Balanced (see chapter 7.2)
- Sentence Similarity Measure: SequenceMatcher
- With Base Rules

- With Longest Consecutive Matching Word Count
- Penalties:
  - Variable-Difference Penalty with Threshold = 0.1
  - No Gap-Size Penalty
- Valid Match Threshold = 0.6

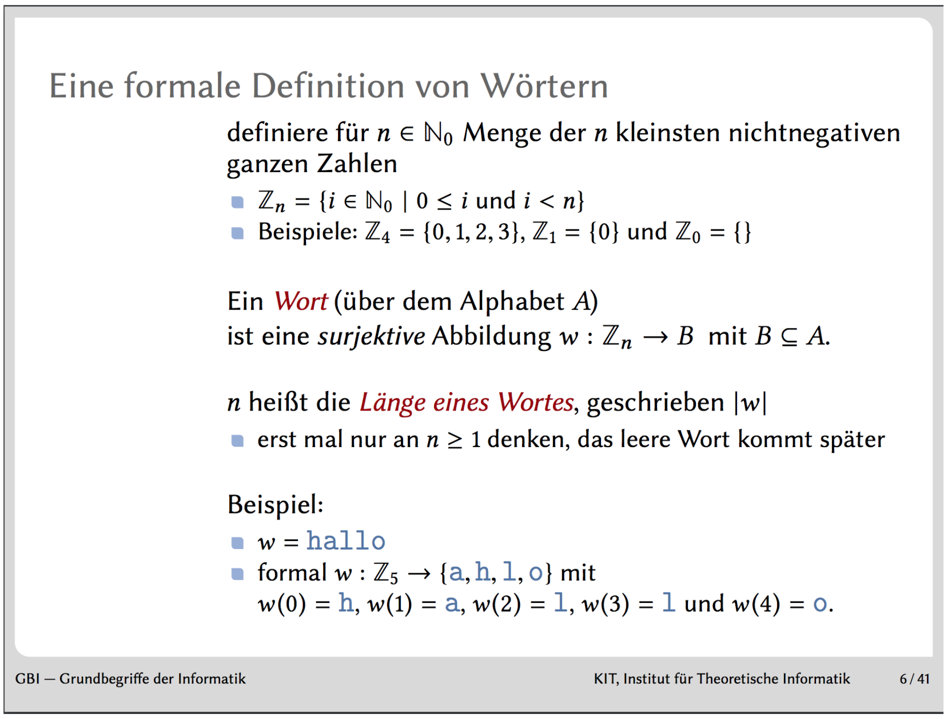
The underlying dataset contains 43 actual formulae.

System Specification	Correct Match Count	Correctly Located Count	Incorrect Match Count
Final System	39	35	11

The system correctly matches 91% of all formulae, correctly matches and locates 81% and only incorrectly matches 11 formulae. For every three correct matches, an incorrect match is expected. This result is sufficiently good to expect an improvement in comprehensibility when the system is used as part of the Lecture-Translator.

The following example shows the system's result for a sentence taken from an actual lecture:

- Lecture slide[1]:



Eine formale Definition von Wörtern

definiere für $n \in \mathbb{N}_0$ Menge der n kleinsten nichtnegativen ganzen Zahlen

- $\mathbb{Z}_n = \{i \in \mathbb{N}_0 \mid 0 \leq i$ und $i < n\}$
- Beispiele: $\mathbb{Z}_4 = \{0, 1, 2, 3\}$, $\mathbb{Z}_1 = \{0\}$ und $\mathbb{Z}_0 = \{\}$

Ein *Wort* (über dem Alphabet A)
ist eine *surjektive* Abbildung $w : \mathbb{Z}_n \to B$ mit $B \subseteq A$.

n heißt die *Länge eines Wortes*, geschrieben $|w|$

- erst mal nur an $n \geq 1$ denken, das leere Wort kommt später

Beispiel:

- $w = \texttt{hallo}$
- formal $w : \mathbb{Z}_5 \to \{\texttt{a}, \texttt{h}, \texttt{l}, \texttt{o}\}$ mit
  $w(0) = \texttt{h}$, $w(1) = \texttt{a}$, $w(2) = \texttt{l}$, $w(3) = \texttt{l}$ und $w(4) = \texttt{o}$.

GBI — Grundbegriffe der Informatik KIT, Institut für Theoretische Informatik 6 / 41

- All LaTeX formulae on the lecture slide:

[1]slide 10 from http://gbi.ira.uka.de/vorlesungen/k-04-woerter-folien.pdf

  - *n \in \mathbb { N } _ { 0 }*
  - *\mathbb { Z } _ { n } = \left{ i \in \mathrm { N } _ { 0 } | 0 \leq i \text { und } i < n \right}*
  - *\mathbb { Z } _ { 4 } = { 0, 1, 2, 3 }*
  - *\mathbb { Z } _ { 1 } = { 0 }*
  - *Z _ { 0 } = { }*
  - *w : \mathbb { Z } _ { n } \rightarrow B*
  - *B \subseteq A*
  - *| w |*
  - *n \geq 1*
  - *w = h a l l o*
  - *w : \mathbb { Z } _ { 5 } \rightarrow { a , h , l , o }*
  - *w (0) = h*
  - *w (1) = a*
  - *w (2) = l*
  - *w (3) = l*
  - *w (4) = o*

- Sentence in German:
  Also Film fünf kann ich also haben als Abbildung von Z5, In, Die, Menge A, H, L, O, und, die, muss.

- Sentence in English:
  So I can film five has to have a mapping of Z5, in the, lot of A, B, L, O, and the..

- Correct formula:
  { Z } _ { 5 } \rightarrow { a , h , l , o }

- Predicted formula:
  { Z } _ { 5 } \rightarrow { a , h , l , o }

- Correct Match?:
  Yes

8. Conclusion

In this thesis, a system was presented that can improve lecture transcripts by recognising mathematical formulae in natural language and matching them with their mathematical representation, which can then be used to supplement the sub-sentences in the transcript that describe the mathematical formulae. This can lead to an improvement in comprehensibility for transcripts with mathematical content.
This system was developed for the use as part of the Lecture-Translator, which is a website for lectures that shows a video recording of the lecturer holding the said lecture, the transcript in the original language, as well as translations of the transcript into multiple languages.

First, a naive system was presented that achieves very bad results. Thereupon, multiple ideas for improvements were presented and evaluated based on empiric observations. The first improvement was to use Base Rules for the classification subsystem that significantly improved the system's results. Afterwards, multiple ideas were presented for the classification and matching subsystems. Those include different classifiers, sentence similarity measures, penalties that make it less likely to match formulae with certain characteristics and values for a valid match threshold that discard all matches below this threshold, which are likely to be false positives.

The final system achieved 91% correctly matched formulae and 81% correctly matched and located formulae on the test data that consists of 148 sentences taken from 4 lectures by 2 different lecturers. It also only matched 11 formulae incorrectly. This makes it suitable to be used as part of the Lecture-Translator and is likely to increase the reader's comprehensibility when reading the supplemented transcript.

Besides improving lecture transcripts, the results of this thesis can be used to pave the way for more research in the area of matching mathematics in natural language with its mathematical representation. Additionally, some results of this thesis can help solve problems that deal with completely different applications but also share the use of a binary classifier or a matching algorithm. These results are the Base Rules for classification and the Variable-Difference penalty.

The next steps are to first incorporate the developed system into the Lecture-Translator and test with the help of many students, whether the system can actually make a difference in comprehensibility. A long-term plan is to replace the rule-based machine translation subsystem with a neural machine translation approach that can support additional languages much more simply. This would require significantly more training data to be labeled in order to properly train the neural net. The classification algorithm would also benefit from more training data, which can lead to an improvement in the overall quality of the system.

Bibliography

[1] Richard Fateman. *How can we speak math?* Tech. rep. 2013. URL: http://www.cs.cmu.edu/usi.

[2] Payam Refaeilzadeh, Lei Tang, and Huan Liu. *C Cross-Validation.* Tech. rep. URL: http://leitang.net/papers/ency-cross-validation.pdf.

[3] Gavin C Cawley and Nicola L C Talbot. *On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation.* Tech. rep. 2010, pp. 2079–2107. URL: http://www.jmlr.org/papers/volume11/cawley10a/cawley10a.pdf.

[4] Jyothi Subramanian and Richard Simon. "Overfitting in prediction models – Is it a problem only in high dimensions?" In: (2013). DOI: 10.1016/j.cct.2013.06.011. URL: http://dx.doi.org/10.1016/j.cct.2013.06.011.

[5] Marc Claesen and Bart De Moor. "Hyperparameter Search in Machine Learning". In: (Feb. 2015).

[6] Hwanjo Yu and Sungchul Kim. *SVM Tutorial: Classification, Regression, and Ranking.* Tech. rep. URL: https://pdfs.semanticscholar.org/cbc3/d8b04d37b2d4155f081cd423380220a91f13.pdf.

[7] Lior Rokach and Oded Maimon. *DECISION TREES.* Tech. rep. URL: http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf.

[8] Leo Breiman. *Random Forests.* Tech. rep. 2001. URL: https://www.stat.berkeley.edu/%7B~%7Dbreiman/randomforest2001.pdf.

[9] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: (). DOI: 10.1145/2939672.2939785. URL: http://dx.doi.org/10.1145/2939672.2939785.

[10] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space.* Tech. rep. arXiv: 1301.3781v3. URL: http://ronan.collobert.com/senna/.

[11] Wikipedia. *Levenshtein distance — Wikipedia, The Free Encyclopedia.* http://en.wikipedia.org/w/index.php?title=Levenshtein%20distance&oldid=864222616. [Online; accessed 20-October-2018]. 2018.

[12] *Jaro distance - Rosetta Code.* URL: https://rosettacode.org/wiki/Jaro%7B%5C_%7Ddistance (visited on 10/20/2018).

[13] David E. Metzener John W. Ratcliff. *JUL88: PATTERN MATCHING: THE GESTALT APPROACH.* URL: http://collaboration.cmc.ec.gc.ca/science/rpn/biblio/ddj/Website/articles/DDJ/1988/8807/8807c/8807c.htm (visited on 10/20/2018).

[14] Kevin Lin and Richard Fateman. *SKEME, (Symbolic Keyboard and Mouse Editor) A Demonstration Program for Multimodal Input of Mathematical Equations.* Tech. rep. URL: https://people.eecs.berkeley.edu/%7B~%7Dfateman/papers/SKEME.pdf.

[15] Cassandra Guy et al. *Math Speak & Write, a Computer Program to Read and Hear Mathematical Input*. Tech. rep. 2004. URL: `http://www.cs.queensu.ca/drl/ffes/`.

[16] Arthur Karshmer, Gopal Gupta, and Enrico Pontelli. *Mathematics and Accessibility: a Survey*. Tech. rep. URL: `http://www.utdallas.edu/%7B~%7Dgupta/mathaccsurvey.pdf`.

[17] *Mathtalk Scientific Notebook and Dragon Naturally Speaking to Dictate Mathematics - YouTube*. URL: `https://www.youtube.com/watch?v=7pjnDIlIjuQ` (visited on 10/20/2018).

[18] Rik Koncel-Kedziorski et al. *Parsing Algebraic Word Problems into Equations*. Tech. rep. URL: `http://ai2-website.s3.amazonaws.com/publications/algebra-TACL2015.pdf`.

[19] Takuya Matsuzaki et al. "Semantic Parsing of Pre-university Math Problems". In: (), pp. 2131–2141. DOI: `10.18653/v1/P17-1195`. URL: `https://doi.org/10.18653/v1/P17-1195`.

[20] Minh Nghiem Quoc et al. *Mining coreference relations between formulas and text using Wikipedia*. Tech. rep. 2010, pp. 69–74. URL: `http://opennlp.sourceforge.net/`.

[21] Kitsuchart Pasupa and Wisuwat Sunhem. "A comparison between shallow and deep architecture classifiers on small dataset". In: *Proceedings of 2016 8th International Conference on Information Technology and Electrical Engineering: Empowering Technology for Better Future, ICITEE 2016* (2017). DOI: `10.1109/ICITEED.2016.7863293`.

[22] Yoon Kim. *Convolutional Neural Networks for Sentence Classification*. Tech. rep., pp. 1746–1751. URL: `http://nlp.stanford.edu/sentiment/`.

[23] James Cannady. *Artificial Neural Networks for Misuse Detection*. Tech. rep. URL: `https://csrc.nist.gov/csrc/media/publications/conference-paper/1998/10/08/proceedings-of-the-21st-nissc-1998/documents/paperf13.pdf`.

[24] Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. "On the Computational Efficiency of Training Neural Networks". In: *NIPS*. 2014.

[25] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. *Applying Support Vector Machines to Imbalanced Data Sets*. Sept. 2004.

[26] Chao Chen and Andy Liaw. *Using Random Forest to Learn Imbalanced Data*. Tech. rep. URL: `https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf`.

[27] David A Cieslak and Nitesh V Chawla. *Learning Decision Trees for Unbalanced Data*. Tech. rep. URL: `https://www3.nd.edu/%7B~%7Dnchawla/papers/ECML08.pdf`.

[28] Zhenyu Wu, Wenfang Lin, and Yang Ji. "An Integrated Ensemble Learning Model for Imbalanced Fault Diagnostics and Prognostics". In: 6 (Feb. 2018), pp. 1–1.

[29] Yoshua Bengio et al. *A Neural Probabilistic Language Model*. Tech. rep. 2003, pp. 1137–1155. URL: `http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf`.

[30] Quoc Le and Tomas Mikolov. *Distributed Representations of Sentences and Documents*. Tech. rep. URL: `https://cs.stanford.edu/%7B~%7Dquocle/paragraph%7B%5C_%7Dvector.pdf`.

[31] Ronan Collobert et al. *Natural Language Processing (Almost) from Scratch*. Tech. rep. 2011, pp. 2493–2537. URL: `http://www.jmlr.org/papers/volume12/collobert11a/collobert11a.pdf`.

[32] Sreelekha S. *Statistical Vs Rule Based Machine Translation; A Case Study on Indian Language Perspective.* Tech. rep. URL: `https://arxiv.org/pdf/1708.04559.pdf`.

[33] Jorge Martinez-Gil. "An overview of textual semantic similarity measures based on web intelligence". In: *Artificial Intelligence Review* 42.4 (2012). DOI: `10.1007/s10462-012-9349`. URL: `https://hal.archives-ouvertes.fr/hal-01630890`.

[34] *GermanWordEmbeddings.* URL: `https://devmount.github.io/GermanWordEmbeddings/` (visited on 10/20/2018).

[35] Andrew P. Bradley. "The use of the area under the ROC curve in the evaluation of machine learning algorithms". In: *Pattern Recognition* 30.7 (July 1997), pp. 1145–1159. ISSN: 0031-3203. DOI: `10.1016/S0031-3203(96)00142-2`. URL: `https://www.sciencedirect.com/science/article/pii/S0031320396001422`.

[36] Takaya Saito and Marc Rehmsmeier. "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets". In: 10 (Mar. 2015), e0118432.

A. Appendix

A.1. Data for Evaluation Graphs

A.1.1. Classifiers

	Training data			Test data		
Classifiers	ROC AUC	Positive Accuracy	Negative Accuracy	ROC AUC	Positive Accuracy	Negative Accuracy
SVC Default	0.802	62.87%	97.54%	0.752	53.24%	97.25%
SVC Tuned & Balanced	0.868	86.51%	87.07%	0.824	78.42%	86.47%
Decision Tree Default	0.897	82.66%	96.82%	0.799	64.03%	95.81%
Decision Tree Tuned & Balanced	0.922	95.62%	88.79%	0.835	79.14%	87.94%
Random Forest Default	0.897	82.66%	96.82%	0.806	64.75%	96.52%
Random Forest Tuned & Balanced	0.922	95.62%	88.78%	0.840	79.14%	88.77%
XGBoost Default	0.981	82.66%	96.80%	0.922	64.75%	96.45%
XGBoost Tuned & Balanced	0.966	90.72%	90.95%	0.907	76.98%	90.96%

A.1.2. Sentence Similarity Measures

System Specification	Correct Match Count	Correctly Located Count	Incorrect Match Count
Levenshtein Distance	20	15	37
Hamming Distance	13	10	44
Jaro Distance	35	32	40
SequenceMatcher	34	30	27

A.1.3. Variable-Difference Penalty

System Specification	Correct Match Count	Correctly Located Count	Incorrect Match Count
No Penalties	37	33	25
Variable-Difference-Penalty 0.05	38	34	17
Variable-Difference-Penalty 0.1	39	35	11
Variable-Difference-Penalty 0.15	37	34	13
Variable-Difference-Penalty 0.2	38	34	11
Variable-Difference-Penalty 0.3	36	32	12
Variable-Difference-Penalty 0.4	32	29	12
Variable-Difference-Penalty 0.5	31	28	13

A.1.4. Gap-Size Penalty

System Specification	Correct Match Count	Correctly Located Count	Incorrect Match Count
No Gap-Size Penalty	39	35	11
Gap-Size Penalty, Threshold = 5, Penalty = 0.01	38	34	13
Gap-Size Penalty, Threshold = 10, Penalty = 0.01	38	34	12
Gap-Size Penalty, Threshold = 2, Penalty = 0.05	37	35	12
Gap-Size Penalty, Threshold = 3, Penalty = 0.05	37	34	14
Gap-Size Penalty, Threshold = 5, Penalty = 0.05	39	35	11
Gap-Size Penalty, Threshold = 10, Penalty = 0.05	38	34	12
Gap-Size Penalty, Threshold = 2, Penalty = 0.1	37	35	11
Gap-Size Penalty, Threshold = 3, Penalty = 0.1	37	34	12
Gap-Size Penalty, Threshold = 5, Penalty = 0.1	38	34	12
Gap-Size Penalty, Threshold = 10, Penalty = 0.1	38	34	12
Gap-Size Penalty, Threshold = 5, Penalty = 0.2	38	34	11
Gap-Size Penalty, Threshold = 10, Penalty = 0.2	38	34	13
Gap-Size Penalty, Threshold = 5, Penalty = 0.3	38	34	10
Gap-Size Penalty, Threshold = 10, Penalty = 0.3	39	35	12

A.1.5. Valid Match Threshold

System Specification	Correct Match Count	Correctly Located Count	Incorrect Match Count
Valid Match Threshold = 0	41	36	33
Valid Match Threshold = 0.5	38	34	22
Valid Match Threshold = 0.55	38	34	18
Valid Match Threshold = 0.6	39	35	11
Valid Match Threshold = 0.65	37	33	13
Valid Match Threshold = 0.7	35	32	10

A.2. List of Translation Rules with covered symbols

1. Summation (Sum, Integral, Product, Co-Product)
   - \sum, \int, \prod, \coprod
2. Function
3. Root
4. Power
5. Fraction
6. Operators
   - =, +, -, <, \leq, >, \geq, \pm, \mp, \times, \div, *, \ast, \star, \circ, \bullet, \cdot, \cap, \cup, \uplus, \sqcap, \sqcup, \vee, \wedge, \setminus, \wr, \diamond, \bigtriangleup, \bigtriangledown, \triangleleft, \triangleright, \lhd, \rhd, \oplus, \ominus, \otimes, \oslash, \odot, \bigcirc, \dagger, \ddagger, \amalg, /, !
7. Greek symbols

  - \alpha, \kappa, \psi, \digamma, \Delta, \Theta, \beta, \lambda, \rho, \varepsilon, \Gamma, \Upsilon, \chi, \mu, \sigma, \varkappa, \Lambda, \Xi, \delta, \nu, \tau, \varphi, \Omega, \epsilon, \theta, \varpi, \Phi, \aleph, \eta, \omega, \upsilon, \varrho, \Pi, \beth, \gamma, \phi, \xi, \varsigma, \Psi, \daleth, \iota, \pi, \zeta, \vartheta, \Sigma, \gimel

8. Trigonometry and special functions
    - arg, cos, cot, csc, deg, det, dim, exp, gcd, hom, inf, ker, lg, lim, ln, log, max, min, pr, sec, sin, sup, tan, arcsin, arccos, arctan, cosh, coth, liminf, limsup, sinh, tanh
9. Accents
    - \hat, \widehat, \tilde, \widetilde, \overline, \bar
10. Numbers
11. Spacing
12. Braces