FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

With the support of the
Erasmus+ Programme
of the European Union

MASTER THESIS

Anna Kriukova

Measuring readability of technical texts

Institute of Formal and Applied Linguistics

Supervisor of the master thesis: Mgr. Cinková Silvie, Ph.D.

Study programme: Computer Science

Study branch: Language Technologies and Computational Linguistics

Prague 2022

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Here I would like to thank those who have supported me throughout working on my thesis.

I would first like to express gratitude to my supervisor, Mgr. Silvie Cinková, for providing valuable feedback on the work in progress, for her advice, and expertise.

I would also like to thank my coordinators from Hyperskill, Darja Orlova and Yulia Akinina, for their attention and guidance that they have given to me over the course of my internship. I appreciate our meetings, which have always left me inspired and full of ideas.

Besides, I would especially like to thank my sister, Alisa, and my friend, Zhenya, who have always supported me and were there for me even from a distance. A special thanks is also to Peter, who has encouraged me in my work, especially on its final steps.

Finally, I wish to show my gratitude to all the LCT students who I have met during this time and whose inspiring examples supported me on this journey.

I must not forget to express my sincere gratitude to the Erasmus Mundus Joint Masters Programme, co-funded by the Erasmus+ Programme of the EU, for letting me be part of this experience.

Title: Measuring readability of technical texts

Author: Anna Kriukova

Faculty of Mathematics and Physics: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Cinková Silvie, Ph.D., Institute of Formal and Applied Linguistics

Abstract: This research explores various approaches to measuring readability of technical texts. The data I work with is provided by Hyperskill, an online educational platform dedicated mostly to Computer Science, where I did my internship. In the first part of my research, I examine classical readability formulas and try to find correlations between their values and the user statistics available for the texts. The results show that there are no high correlations, thus, the standard formulas are not suitable for the task. The second part of the research is dedicated to experiments with machine learning algorithms. Firstly, I use four sets of features to predict the average rating, completion time, and completion rate of a step. Then, I introduce a rule-based algorithm to split the texts into well- and poorly-written ones, which relies on students' comments. However, binary classification trained on this division shows low results and is not used in the final pipeline. The system suggested as the outcome of my work employs the user statistics' prediction for new texts and a rule-based comment-focused algorithm for the published texts. As a result, texts that the system identifies as "bad" are reported to the Hyperskill content team for improvement.

Keywords: readability technical texts data analytics corpus linguistics comprehensibility machine learning

List of Figures

List of Tables

Contents

Introduction

This thesis is based on my internship at Hyperskill, an online educational platform that promotes learning through hands-on projects. Most of the content on Hyperskill deals with different areas of Computer Science, which defines the technical nature of the texts on the site.

Overall, the mission of my work is to assist Hyperskill in their aim to have outstanding content. In my research I explore various ways of measuring readability in order to identify less readable texts on the site. As a result, the content team can review published texts to improve them if needed, and check how readable texts are before publishing.

The task of measuring readability of texts has lied in the intersection of linguistics, teaching methodology, and, later, NLP for almost a century. It started in the 1920s with the research of Lively and Pressey [1923] that introduced what is believed to be the first readability formula. The methods of approaching the problem have changed through the years, but it still, remains a challenging and presently topical task.

The aim of readability assessment is to predict whether a text's audience will understand its content, which is, conceptually, a universal goal of textual communication.

The interest in this topic is emphasized, for example, by the existence of a plain language association for the English language (US, UK). It promotes understandable writing, especially on governmental sites, and formulates the guidelines on how to achieve it. It also takes on the task of consulting governmental organizations and teaching people to write more understandable text. A work, similar in purpose but different in area, was done for French legislative texts François et al. [2020]: it estimates the readability of such texts and suggests ways to help writers make them better.

The need for specialized formulas for various text styles was highlighted by Dell'Orletta et al. [2011] who confirmed that a readability model can only correctly assign labels to the same genre of texts it was trained on. Among the first research on the readability of specialized texts were Jacobson [1965] or Shaw [1967] working with scientific texts. Other instances include Hull [1979] for technical texts, and Sheehan et al. [2013] for "informative" and "literary" texts. In my work, I incorporate several features specific to the texts on computer science in general and to the texts on the Hyperskill platform in particular.

Another, more recent use of readability formulas is connected to evaluation of NLP systems or controlling the difficulty of their output. For example, Marchisio et al. [2019] train machine translation system on corpora with different readability to teach the system to produce an output of the specified reading level. Working with explainable recommender systems, Costa et al. [2018] compute several readability scores for automatically generated explanations and try to make the text more natural by modifying it to match the readability level of the existing reviews. A text simplification system described in Aluisio et al. [2010] receives an output from the text author on how much they want to simplify their text, and does it basing on the readability levels.

The assessment of readability is also used for educational purposes. For ex-

ample, to readjust textual materials to the level of the reader Begeny and Greene [2014], or Stenner [1996], where the researchers built an education tool to match readers with books. Similarly, readability is often used in the second language studies where texts are ranked according to the language acquisition level (Heilman et al. [2007], Vajjala and Meurers [2012], Shen et al. [2013]).

Another interesting direction of readability in education deals with assessing readability of course materials, the idea behind which is to make the level of texts optimal for the audience. Klare and Smart [1973] analyzed the readability of printed correspondence courses used by the military. They showed that some courses were too difficult for readers with the average reading skill and found a high Spearman rank correlation between the readability score and the probability of students completing the course. This emphasizes the importance of measuring the readability of educational materials.

Summing up, the research shows that measuring readability usually pursues the following objectives, be it with regular texts or automatically generated ones:

- Estimate how understandable a text is, taking into account its purpose and genre.
- Modify the text to match the level of the audience.

My work complies with these objectives but essentially, my goal is more practical: provide the Hyperskill team with a process to identify if their educational content is suitable for the audience in terms of its clearness and identify cases when it's not.

The rest of my work is structured as follows:

- In chapter 1 I discuss how readability was defined throughout research in this field and formulate my own approach to its definition. I also provide a brief overview of the history of readability studies.
- Then, in chapter 2 I thoroughly describe the data I work with.
- Chapter 3 is dedicated to experiments on the correlation of classical readability scores with various user statistics of texts.
- Chapter 4 explores the use of machine learning models for the regression tasks of predicting three user statistics of texts and for the binary classification task of identifying whether a text is understandable enough.
- Finally, I sum up the work and the main results in the Conclusion.

1. Research on readability

1.1 Definition of readability

There are a lot of various approaches to readability. The one in which most of researchers agree (François [2015b]) is that readability considers the audience as a whole, not individual users. The reading experience of a particular person is not taken into account, the audience is considered to be homogeneous.

Another point that everyone seems to accept is that readability cares about text comprehension. There are various ways to measure it, but in general, this criterion is taken as the most important one (Dubay [2004]).

However, there are many other aspects that are sometimes included in the concept of readability. One that is generally acknowledged to matter for readability is reading speed (Dubay [2004], Mcclusky [1934], Boudjella et al. [2017]). Usually, it is difficult to trace it, but in my work, I will operate with this parameter, since it is logged on the website.

Sometimes, people's interest in the topic is also considered as part of readability Entin [1981], Klare [1976], Fass and Schumacher [1978]. However, it might be a very difficult parameter to estimate. In my work, I do not take it into account, assuming that students learning Computer Science online are motivated to complete the course and this parameter can be taken out of the equation.

Readability can be seen as including even more additional aspects, such as the design of the page (layout, font type and size, colors) (Hill and Scharff [1997], Hussain et al. [2011]). Indeed, these attributes can affect readability, but the current research on this topic is quite narrow. What is more, in many settings when we are concerned with readability, these characteristics are pre-defined, and what can be altered is the textual content itself. This is the case in my study, and I do not consider these aspects as a part of readability.

More standardly, readability is seen only from the perspective of style, and it is regarded separately from the questions of content, coherence, and organization (Dubay [2004] citing Klare [1963]). However, in the research of Klare et al. [1955] it was shown that style is more important for people without prior knowledge of the material, otherwise, people profit little from the change of style. A similar hypothesis was confirmed in Entin and Klare [1985], according to whom easier readability of a text was more beneficial for those of less knowledge and interest in the topic. Thus, though the style is classically considered very much connected to readability, sometimes, what proves more important is the prior knowledge of the audience.

A similar issue with measuring readability is how it is related to the conceptual difficulty of the text: a well-written text about a complex concept can be hard to understand not because of the style, structure, or format but because of the topic itself. Sometimes, no distinction is made between readability and text difficulty (Zakaluk and Samuels [1988]). In a throughout research of Gray and Leary [1935], authors experiment with various parameters taken from four groups: content, style, format, and organization. Their results show that content is even slightly more important than style, and format and organization influence readability approximately to the same degree, but less than content and style.

Based on the mentioned ways to approach readability, I may characterize the one I use in my research:

1. I consider the audience as a whole rather than focusing on each particular user.

2. I use several acknowledged criteria as a proxy of readability:

   - Reading speed, which is available because the approximation of this metric is measured on the platform I work with.
   - Text comprehension, regarded as a binary feature and dependent on the comments that students leave for texts (see chapter 4). Due to the specificity of the problem, I could not find research that would implement a similar approach, but the explanation behind this choice is the following: if there are comments on the texts and several students write about their dissatisfaction with the material in terms of the language or understanding, we believe that this text is written worse than the one without negative comments.

3. Experiment with two other user statistics as a proxy of readability (see chapter 3):

   - the average rating of the text from users;
   - the completion rate, meaning how many students finished the text after opening it.

   These metrics are specific to the setting of my work and, to the best of my knowledge, there is no research on text readability that would use such parameters. The underlying rationale for their use is logical: if a text is not understood by a lot of students, it will not receive a high rating. Similarly, if a topic is finished by a small proportion of people, it is likely to be too difficult.

4. I do not regard the motivation of users as a part of readability and consider it to be equal among all students.

5. I do not take into account the layout of the page either (more about it in 2).

6. While representing readability, I try to address text style, structure, and conceptual difficulty, as well as prior knowledge of students through relevant features.

1.2 Development of readability studies

The research on readability dates back to the beginning of the 20th century, when linguists for the first time started assessing comprehensibility of texts. In the earlier stages of research, it was driven by two main areas: either readability for schoolchildren (Spache [1953], Smith [1961], Kintsch and Vipond [1979]), or for the US army (Kincaid et al. [1975], Kern [1980]). From the beginning, the main

instruments for measuring readability were various readability formulas based on surface text characteristics.

The first readability formula is considered to be Lively and Pressey [1923] where authors analyzed three text features: 1) the number of different words 2) the percentage of words that do not occur in the list of generally known words from Thorndike [1921] 3) a weighted median of ranks of all words in this list. Soon after that, a classical procedure for creating a formula was developed (Vogel and Washburne [1928]). It involved the choice of features to capture various aspects of readability, the choice of evaluation criteria, and the search for coefficients that would maximize the correlation of the formula results with the evaluation criterion. Examples of such standard readability metrics are Kincaid et al. [1975], Spache [1953], Coleman and Liau [1975], Smith and Senter [1967], McLaughlin [1969], Chall and Dale [1995], which I describe in more detail later.

In the 60s, the possibility to automate the readability formula search appeared. Smith [1961] is believed to be the first work in this direction, and it led to the use of even more features. Thus, Daoust et al. [1996] as cited by François [2015b] automatically extracted 120 linguistic variables and trained multiple linear regressions to reach the best results.

The evaluation criteria mentioned as a part of creating a readability formula could be taken from various sources:

1. From expert judgements (e.g. Lively and Pressey [1923], Vogel and Washburne [1928]).

2. From reading tests, such as comprehension questions or MCQ (e.g. Vogel and Washburne [1928], Ojemann [1934]). Usually 75% correct scores on a multiple-choice test is set as the criterion for optimum difficulty as Dubay [2004] refers to Thorndike [1916].

3. From cloze tests. Cloze tests were initially introduced by Taylor [1953] and became a standard way to measure reading comprehension. They ask readers to fill in the missing words in a text (usually every fifth word) and take the percentage of the correct values as the score.

After the continuous work on standard readability metrics, in the early 80s, a new paradigm was introduced. The classical readability formulas started being criticized as too shallow, and the need for a new approach was expressed Kintsch and Vipond [2014]. As a consequence, new features were taken into account, many of them psycholinguistically grounded (Williams et al. [1977]). Researches experimented with such metrics as text organization (Armbruster [1984]), discourse cohesion (Clark [1981], Kintsch [1979], McNamara et al. [2014]), inferential load (Kintsch and Vipond [1979], Kemper [1983]), and rhetoric structure (Meyer [1982]).

However, these features, being more difficult to implement, did not prove to bring substantial improvements. Several experiments show that their results are similar to those of classical formulas or that the contribution of such features is not that evident (Kemper [1983], Schwarm and Ostendorf [2005], Crossley et al. [2007], Bormuth [1966]). It does not prove that their use is unjustified, but the main conclusion is that the standard readability metrics, though grounded

on surface text parameters, are usually capable of capturing text readability as measured by comprehension tests (Dubay [2004]).

The latest approaches, like in other NLP areas, mostly use statistical models to predict readability. The examples of work in this direction include Si and Callan [2001], probably the first approach to readability as a classification problem, Tanaka-Ishii et al. [2010], where authors regard it as a sorting problem, and Azpiazu and Pera [2019], where the input of the system is just raw text, and an RNN with attention predicts readability in a multilingual setting. However, many such models, especially the ones using neural networks, require large training corpora. This is the reason why they are not applicable in my research: the size of the data is only around 1000 instances. Some classical machine learning techniques, nevertheless, can be used, which I also experiment with in my work (see chapter 4).

2. Data description

This chapter introduces the main concepts that I operate with through my work and describes the data: texts themselves, their distribution and user statistics, as well as comments and their characteristics.

2.1 About Hyperskill in more detail

On Hyperskill, all educational materials are represented in textual form. One of the main content units is a *topic*; it consists of a theory part and several tasks that test students' understanding of the material.

- The **theory** part is basically a short theoretical article that is devoted to a particular concept, e.g. Python lists, introduction to transactions in databases, or z-index in CSS positioning.
- The **tasks** vary from multiple-choice questions to code problems, but they are usually concise and ask to answer a specific question/do a concrete thing.

In the current work, I focus only on theoretical parts of topics because this is the main educational unit on Hyperskill that introduces new material, and its quality and understandability are essential for the platform. What is more, tasks often consist of only several sentences formulating the question, and the readability of short texts has been acknowledged to be a much more challenging task (Bormuth [1966], Lenzner [2014], Dell'Orletta et al. [2011]).

It is worth mentioning that topic creation is standardized, and there are certain criteria that each topic should follow. These criteria deal with the following aspects:

- Topic structure: the number and size of sections, introduction, and conclusion.
- Formatting: the use of bold and italic fonts, adding pictures, links, and tables.
- Language: guidelines about tone and style, recommendations to make writing better. The latter include, for example, reminders to break down complex sentences, choose active voice over passive, and vary the vocabulary. The former ask authors not to use technical terminology or jargon, avoid documentation style, stick to B2-oriented English, and other similar guidelines.

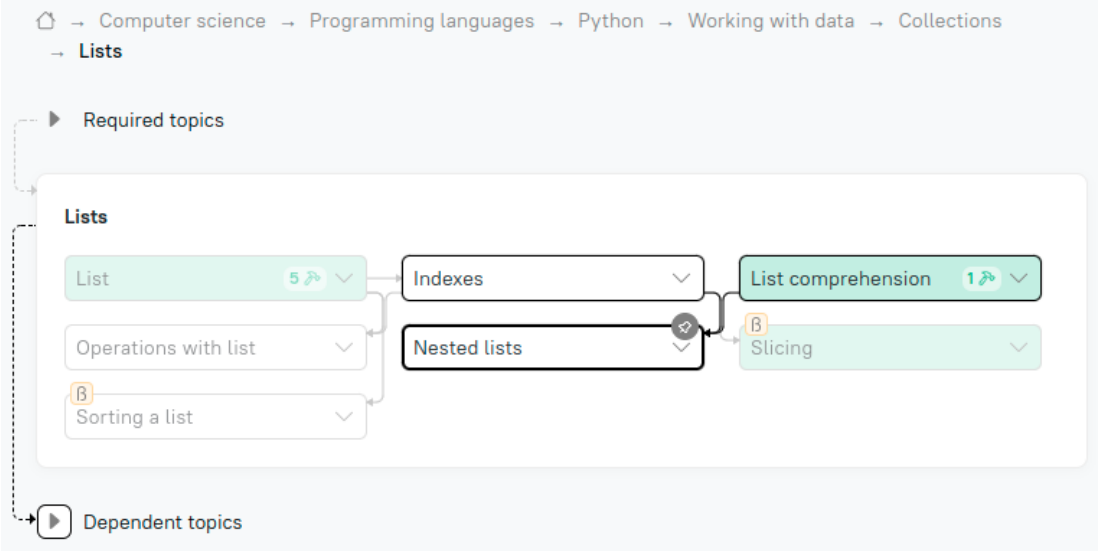
All of these criteria are dedicated to increasing texts' comprehensibility.

Apart from that, the design of the texts on the site is standardized too: such parameters as the font, its size, and the layout of different parts on the page are pre-defined and cannot be altered by content makers.

All this makes the texts presented on the site more homogeneous, which is an advantage for my work. First of all, we don't need to take into account meta-factors, such as the overall design of the page. Besides, there are certain language guidelines that authors should follow for their topics, and all materials go through proofreading before publishing. Thus, we can be sure that what makes one topic more readable than the other is not the language register or style but rather the concept difficulty and specific language choices of the author.

Another characteristic feature of Hyperskill is that the study path of a user is non-linear. It means that the order of topics that a student learns may vary. Instead of a linear sequence of topics, they are represented as a knowledge map: a general graph of topics is separated into groups, and each topic has its own dependencies. In figure 2.1 you can see a group of topics dedicated to lists in Python, and that the topic "Nested lists" depends on two other topics: "Indexes" and "List comprehension".

Figure 2.1: Illustration of the knowledge map and prerequisites



These dependencies are called prerequisites, and students are expected to complete all prerequisites of a topic before reading it. In further sections, I get back to the concept of prerequisites: information about them is used for customized vocabularies for the Dale-Chall readability formula, as well as in the machine learning systems as features.

Finally, I would like to emphasize why I do not use tasks as a means of assessing the readability of texts. On the one hand, they could be regarded as comprehension tests, a standard approach to measure how well a student understood the text, especially since the information on user performance is available. On the other hand, there are considerable potential problems:

- The tasks on the platform were not created specifically to assess the readability of the texts; they may ask not about the main points of a text but rather about a small but important detail, which would not provide information on the **readability** of the text;

- There are various types of tasks, such as code problems, multiple choice questions, sorting problems, etc; with the type, not only the type of possibly asked information varies but also the acceptable ranges of errors (e.g. in code tasks students, in general, make more mistakes before finding the correct answer).
- Another general problem of using tasks for evaluating readability, according to François [2015a], is that the vocabulary of the questions affects the user performance.

Taking these into account, I decided to ground the readability measurement of texts on other criteria that will be explained further in this chapter.

2.2 Data extraction

The data that I had access to initially contained texts in the XML format. Therefore, the first thing that I needed to do was extract plain text from it. An example of the XML texts is in figure 2.2.

Figure 2.2: Example of row texts

	A	B
1	step_id	text
2	18996	" The example below contains two variables: \n\n \n\t <code class=\"language-go\">source</code> — string to search
3	18684	" Chose the correct statement about the <code class=\"java\">ColorSpace</code> class. "
4	12043	" Find a derivative of the function <span class=\"math-tex\">\\(G(x) = \\sin (x^2)\\). "
5	9670	" What would be the result of the expression below? \n\n<pre><code class=\"language-scala\">val userList = List(\n (\"Alice\"
6	13771	" \n\n Find the <span class=\"math-tex\">\\(x\\) value of the local minimum for <span class=\"math-tex\">\\(f(x) =
7	18683	" Which of the following statements concerning grayscale transformation are correct? "

The way of extraction was a standard one but there is a detail that is worth mentioning. In the texts about programming languages, as well as some others, there are often code snippets inside the text. Since I don't want to consider parts of code the same way as I do with regular texts, I substituted them with the special token **code**. I did the same with the images, mathematical formulas, and tables. The code for this task and all the following ones is provided in my GitHub repository[1].

2.3 Texts description

In this section, I describe the data I used in my research. As mentioned previously, I work with the theoretical parts of the topics. Apart from the texts themselves, I had access to their user statistics as well as comments that students leave under theories.

The following part of this section is structured as follows: firstly, I talk about the main characteristics of the texts, then I describe their statistical features from users, and finally, I mention the comments dump that I used.

The initial data dump that I used included topics (theory steps and tasks) and their user statistics as of the date of 6 April 2022. There were 13026 steps

[1]https://github.com/kr-ann/re-ada

in general, among which 1321 (10%) were theories and 11705 (90%) were tasks. They belong to 8 various areas: Maths, Fundamentals, Programming languages (Java, Kotlin, Python, JavaScript, Golang, Scala), Mobile, Frontend, Backend, Data Science, and Desktop.

Since I am working only with the theoretical steps, my initial dataset contained 1321 texts. It, however, required some cleaning, which I describe in the next section. There were 1064 texts left after the cleaning, and in the table 2.1 below you can see their main statistics.

	Overall	Avg (per theory)
#sentences	34773	32.68
#tokens	790823	743.25

Table 2.1: Basic statistics of the data

To work with all data dumps throughout the research I make use of `numpy` (Harris et al. [2020]) and `pandas` (McKinney et al. [2010]).

2.4 Texts' statistics

Along with topics' texts, I got access to some statistical metrics that are tracked on the site. All of the metrics can be influenced by the theory step's readability, that is why it is worth taking them into account:

- seconds_to_complete – how many seconds it takes for a user to complete the theory step (from opening it till they proceed to the tasks), median value across the 200 most recent completions. Basically, it is the time that a user needs to go through the text and understand it. Such a parameter of reading time is often said to reflect the readability of a text Boudjella et al. [2017]. However, since all theories differ in size, to be able to compare this parameter across different texts, I normalized it, receiving the number of seconds it takes for a student to read 10 tokens from the text. It also proved beneficial since some outliers in terms of the initial seconds_to_complete were no longer outliers after normalization.

- completion_rate – a completion rate of the step for the last three months; ranges from 0 to 1 and corresponds to the ratio of users who completed the step to those who opened it. It describes what part of the users gets to finish the text; if it is low, it can signify that the text is difficult for the target audience.

- completed_step_users_count – how many users completed the step. This parameter is needed to account for differences that topics' metrics have because of the number of users who finished it. I use this metric for further filtering of texts.

- avg_like – average rating of the step, ranges from -2 to 2. As a user's rating, it can demonstrate whether students considered the content good. After each topic, there's a scale with 5 emojis, from the angry face (corresponding to -2 on the scale) to a happy one (corresponding to 2), and then the evaluations are averaged.

- likes_count – the number of ratings the step has (equal to the number of users who evaluated it); similarly to completed_step_users_count, it can be used to account for differences between average likes for various topics and is used for filtering the data.

- topi_completion_rate – shows what part of students who tried to complete any of the topic's stages (theory or tasks) actually completed the topic. Usually, students read the theory part first, and to complete the topics, they need to pass a number of tasks after it. Thus, this metric can reflect how easy it was for the audience to finish the problems after reading the theory.

- completed_topic_users_count – how many users completed the topic: similarly to counts of users for theory completion and rating, it can influence how we regard the previous metric of topic completion rate.

- back_to_theory_times_per_user_session_avg – how many times per session the average user got back to the theory of the current topic while solving the tasks. Intuitively, the more times a student needs to re-read parts of the theory, the harder it was for them to understand it from the first time. "User session" here is the time since the student starts reading a theory, until 30 minutes after the last action in the topic's steps (theory or tasks). This metric, however, cannot capture cases when a student solves tasks in one tab and has an opened theory in another tab: Hyperskill keeps track of active actions on the page, but the page viewing is not logged.

- back_to_theory_users_% – percent of users who got back to the theory while solving the tasks. Similarly, a higher number could mean that it was a more difficult theory text.

The distributions of these metrics on the data are shown in A.3.

I had an option to choose the period for these statistics: three months or a year. I preferred the former since it captures the more recent status of content. It is worth saying, however, that some of the texts could have been changed in these three months: for example, if a theory is rewritten and made more clear, it starts receiving "better" statistical metrics, whereas the average statistics may not change fast enough to reflect this shift. However, texts are not changed often, and we cannot trace such cases, so I consider the existing statistics to objectively reflect the students' impression of the topic at the current moment.

As mentioned in the previous section, the initial theoretical steps required some cleaning. It included deleting entries that had NaN values, and those that were completed by less than 20 people. The latter decision is explained by the fact that such steps' statistical metrics would not be stable enough to take them into account.

The number 20 was also used when I analyzed the average like parameter: on our data, the avg_like metric becomes more stable after a step received 20 evaluations. This has been found by analyzing the graphs in figures 2.3 and 2.4. I've taken popular theory steps (from topics that were completed by >150 students in the last 28 days) and their evaluations for the previous three months. Next, on this data, two graphs were built: the first one showing the mean rating of

the step, and the second one depicting the standard deviation of the ratings, and how the mean rating and the STD change with each new evaluation. According to the graphs, after around 20 evaluations, both the average rating and its STD more or less stabilize.

Figure 2.3: The change of mean avg like with the increasing likes count

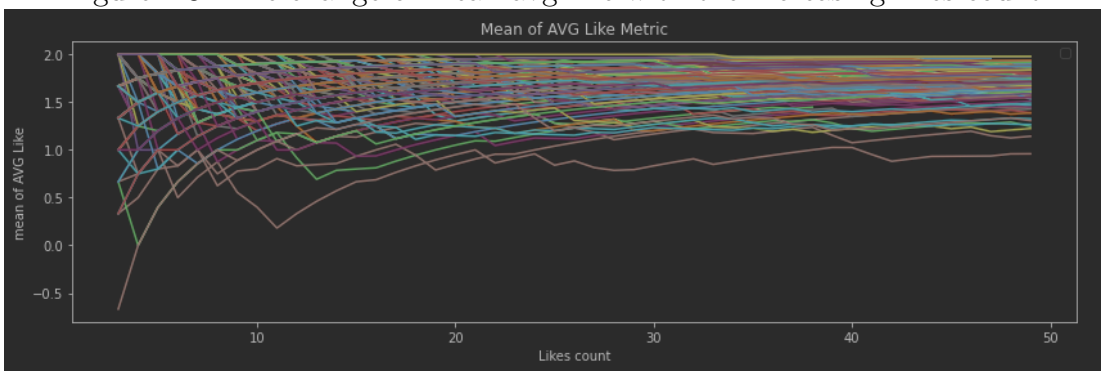
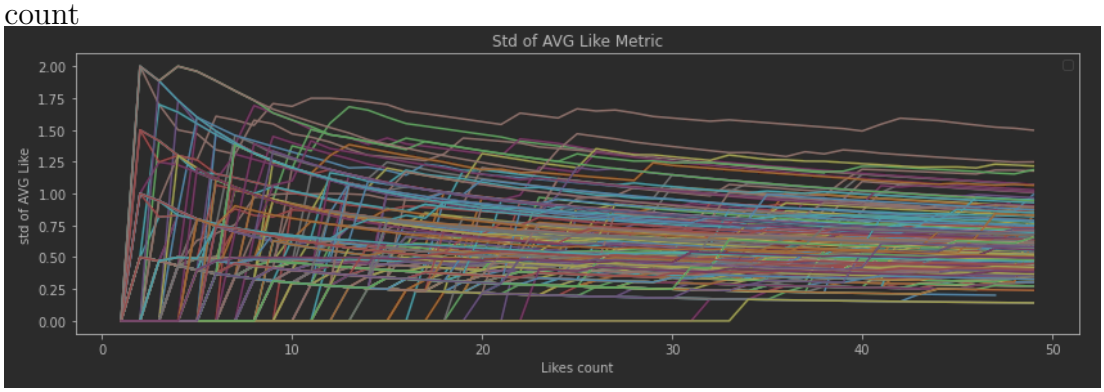


Figure 2.4: The change of the std of the avg like metric with the increasing likes count



It is also worth mentioning that having a smaller threshold for the number of likes is beneficial: the higher the threshold, the fewer topics have the needed number of likes, and so the smaller the corpus is.

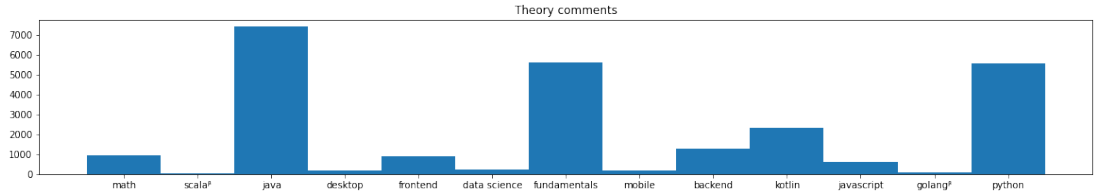
2.5 Texts' comments

Other data that I had was a dump of comments for the theory steps. The distribution of comments across various areas is not uniform, as can be seen in figure 2.5.

Java, Fundamentals, and Python have significantly more comments for theoretical steps than other areas. This can be explained simply by the fact that there are more topics belonging to these areas than to others. However, this should not influence my work since out of 1321 theoretical steps, most of them, namely 1120 (85%), do have comments.

For each comment, except for the comment's text itself, the dataset included several fields with information about the comment. For my research, I needed

Figure 2.5: The distribution of comments across various areas



only two of them: the number of positive reactions the comment received, and the comment's "status". The status was one of "new", "pending", "fixed", and "won't fix", which denotes the stages of the work with comments:

- "won't fix" means that the comment was not found useful by the content team to make improvements to the topic's text;
- "fixed" denotes that the comment was taken into account, and the topic was altered thanks to the comment; such comments are automatically hidden from the site.
- "pending" is the state when the comment is being checked by the content team;
- "new" refers to comments that either haven't been checked before or those that don't contain any information that could be used to improve the theory.

The comments' "status" I use later in the research to filter out comments that were "fixed" and therefore their corresponding theory step does not have the problem mentioned in the comment anymore. The number of positive reactions to the comment I take into account when counting the number of "bad" and "good" comments the theory received. This is explained in more detail in further sections.

2.6 Summary

In this chapter, I introduced the main unit of the Hyperskill platform: theoretical topics. I mentioned that they obey certain rules in terms of structure, formatting, and language, which makes them more or less homogeneous. Then, I illustrated the concept of the knowledge map and prerequisites: dependencies that each text on the platform has and that the students must complete before reading the current material. Next, I outlined the algorithm of data collection and filtering and described the user statistics that were available for each text. Finally, I mentioned the dump of comments and its main characteristics, namely "status" of a comment and the number of positive reactions. The next chapter is dedicated to data analysis and experiments with the standard readability metrics.

3. Correlation experiments

This chapter describes the first out of the two main parts of my research. It analyzes the data and explores the standard readability metrics by tracing their correlations with the user statistics of texts.

3.1 Introduction of the scores

As mentioned before, there are guidelines for topics' theories that provide for more homogeneity between various texts. Some of these guidelines that can be checked automatically are traced by a specially developed tool. It is called a *validator* and content authors on Hyperskill use it before publishing the topic. It considers such aspects as the number and length of the sections, misspelled words, correct formatting of headers, and others. Along with these, the validator also shows two readability scores: the **new Dale-Chall readability formula** Chall and Dale [1995] and the **Flesch reading ease score** Flesch [1948]. These scores, however, are not analyzed by Hyperskill team members when uploading a text to the site, mainly because they were not proven to work well for Hyperskill texts, and there were no ranges of recommendable values for them. My first task, therefore, was to explore their results. As a criterion to measure how suitable a readability formula is for the platform's texts, I chose the presence of correlations between the scores and topic statistics, especially the normalized reading time. If the correlations turn out to be significantly high, the results of the readability formulas could be used to make predictions about future text statistics before publishing the topic.

Flesch Reading Ease formula is based on two parameters:

- Average sentence length (ASL);
- Average number of syllables per word (ASW).

$$Score = 206.835 - 1.015 * ASL - 84.6 * ASW$$

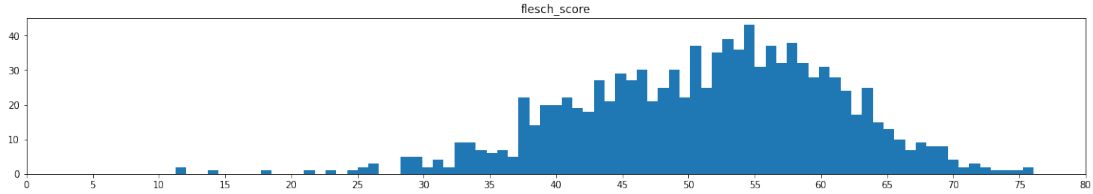
The result is a number on the scale from 0 to 100, where the higher the score is, the easier the text is to understand. A score below 30 is considered very difficult and a score around 70 is seen as suitable for adult audiences.

Figure 3.1 illustrates the distribution of scores on our data. The texts with scores between 60 and 70 are thought to be suitable for 13-15-year-old students, the ones between 50 and 60 are considered to be fairly difficult, and the ones between 30 and 50, where many texts lie, are said to be understood by college graduates.

New Dale-Chall readability formula Chall and Dale [1995] also considers two parameters:

- Average sentence length (ASL);

Figure 3.1: Distribution of the Flesch Reading Ease scores on our data



- Percent of "difficult" words (PDW), those that are not in a vocabulary of simple words. Even though the vocabulary was created in 1984 and revisited in 1995 which means that it is not up-to-date, this formula has been consistently mentioned in research and proven to be good for measuring readability Dubay [2004], which is why I use it here.

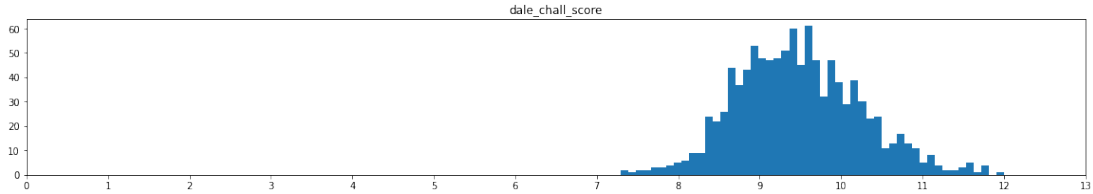
$$Score = 0.1579 * PDW + 0.0496 * ASL$$

$$If PDW > 5\% : Score+ = 3.6365$$

The result of the score is the grade level for a reader who can answer half of the questions for a text.

Figure 3.2 illustrates the distribution of the scores on the dataset. Most of the Dale-Chall values lie between 8 and 11, where the units 8-9 correspond to grades 11-12, the units 9-10 correspond to college students, and the units 10 and above correspond to college graduates.

Figure 3.2: Distribution of the Dale-Chall scores on our data



3.2 Correlations with the initial scores

To compute correlation, the first thing to do was to receive the readability scores for each text. To have more control over how they are calculated, I used the library `py-readability-score`[1] (DiMascio [2019]) instead of taking them from the validator.

Following this, I computed Spearman rank correlations between all statistical metrics mentioned in section 2.4 and the two readability scores. The results can be seen in table 3.1. The p-value was rounded to 5 decimal places, so zeros in the table mean that it was even smaller.

[1] https://github.com/cdimascio/py-readability-metrics

	Flesch		Dale-Chall	
	score	p-value	score	p-value
normalized seconds to complete	-0.186	0	0.177	0
completion rate	0.099	0.0011	-0.074	0.01601
avg like	0.009	0.82518	-0.104	0.01005
topic completion rate	0.025	0.4138	-0.009	0.77102
back to theory times per user session avg	-0.096	0.00176	0.087	0.00438
back to theory users %	-0.116	0.00014	0.088	0.00419

Table 3.1: Correlations between the Dale-Chall and Flesch scores and all user statistics of texts

The highest correlation for both scores is with the normalized "seconds to complete" parameter, and they are statistically significant (p-value <0.00001 for both). However, the coefficients are still really low: -0.186 and 0.177, so in this case, the readability scores cannot serve to predict the time it takes to complete a topic's text.

The correlation of the two readability scores themselves, however, is predictably high: -0.67. The value is negative because in the Flesch formula, the higher the score, the more readable the text is considered to be, whereas for Dale-Chall it is vice versa.

3.3 Correlations of the statistical metrics between each other

Other correlations that I intended to check were correlations of texts' statistics between each other (table 3.2).

The highest correlation between two scores is 0.97, the one between two parameters corresponding to re-reading the theory while solving tasks: "back to theory users %" and "back to theory times per user session avg". This is an expected result since they describe very similar concepts. One of them, "back to theory users %", also correlates well with other metrics, except for the average like and the topic completion rate. The same situation holds with the normalized seconds to complete: it has rather high correlations with all metrics except for the mentioned two. The average like and the topic completion rate don't show any significant correlation with other parameters, so they cannot be predicted based solely on other metrics. However, for predicting the rest of the metrics, "seconds to complete" and "back to theory users %" can be used.

3.4 Correlations with other readability scores

The two scores I started my experiments with – Dale-Chall and Flesch ones – were chosen because they are implemented in a validator that checks texts on Hyperskill before they are published. However, there exist a large number of other classical scores, and it was interesting to see whether their results would

	completion rate		**avg like**		**topic completion rate**	
	score	p-value	score	p-value	score	p-value
normalized seconds to complete	-0.494	0	-0.042	0.29742	0.012	0.77614
completion rate	1	0	-0.023	0.57693	-0.101	0.01251
avg like	-	-	1	0	-0.002	0.95814
	back to theory times		**back to theory %**			
	score	p-value	score	p-value		
normalized seconds to complete	0.687	0	0.669	0		
completion rate	-0.463	0	-0.51	0		
avg like	-0.013	0.74929	-0.008	0.8498		
topic completion rate	0.005	0.90796	0.091	0.02436		
back to theory times per user session avg	1	0	0.97	0		

Table 3.2: Correlations of statistical metric between each other

differ from the Flesch Reading Ease and Dale-Chall formulas. The library that I used for the two previous scores[2] implemented seven other standard scores, so I decided to stick to them. I wanted to check their correlations with the texts' statistics as well: to see if any of them allow predicting texts' metrics better than Dale-Chall or Flesch scores.

I experimented with the following scores:

- **Flesch-Kincaid** Kincaid et al. [1975]. It is similar to Flesch reading ease but supposed to be an improved version; it's based on the same parameters as the Flesch score.

$$Score = 0.39 * ASL + 11.8 * ASW - 15.59$$

- **Spache** Spache [1953]. Developed for evaluation of texts for the 3rd-grade level or below. It's similar to Dale-Chall but has different coefficients and another list of simple words.

$$Score = 0.141 * ASL + 0.086 * PDW + 0.839$$

- **Gunning-Fog** Gunning [1952]. Like many other scores, it takes into account the average sentence length. The second parameter, however, is different from all other metrics: it is the percent of hard words, and "hard words" are considered those of three or more syllables that are not proper nouns, combinations of easy words or hyphenated words, or two-syllable verbs made into three with -es and -ed endings.

$$Score = 0.4(ASL + PHW)$$

[2]https://github.com/cdimascio/py-readability-metrics

- **Coleman-Liau** Coleman and Liau [1975]. It was developed for technical texts, therefore it usually gives them a better readability grade than other formulae. It is based on the average number of letters per 100 words (L) and the average number of sentences per 100 words (S).

$$Score = 0.0588 * L - 0.296 * S - 15.8$$

- **Automated Readability Index** Smith and Senter [1967]. Except for the average sentence length, this score uses the average word length, taken in characters.

$$Score = 4.71 * AWL + 0.5ASL - 21.43$$

- **Linsear-Write** O'hayre [1966]. This formula is built on the number of easy words (EW) that are words of one or two syllables, the number of hard words (HW) consisting of three or more syllables, and the number of sentences (S). In the end, the score is adjusted based on its intermediate value.

$$Score = (EW + HW * 3)/S$$

$$IfScore > 20 : Score = Score/2$$

$$IfScore <= 20 : Score = (Score - 2)/2$$

- **Smog** McLaughlin [1969]. Just as the previous score, it relies on the number of polysyllable words (PSW), that is words with three or more syllables.

$$Score = 3 + \sqrt{PSW}$$

The table 3.3 illustrates the correlation results for all of the mentioned readability metrics.

Interestingly, the two metrics that did not show correlations with other metrics in the previous section, topic completion rate and avg like, are the ones that don't have significant correlations with any of the readability scores here either.

Similar to the results for Dale-Chall and Flesch readability scores, the best correlations among the text characteristics are for the normalized seconds to complete parameter. The highest one is for the Coleman-Liau score, 0.198.

In contrast to Dale-Chall and Flesch scores that had visible correlation only with the seconds to complete metric, some scores of the newly implemented metrics have significant correlations with two or even three parameters: Spache (3 correlations) and Coleman-Liau (2 correlations). Three other scores, on the contrary, do not correlate with any of the statistical metrics at all (Gunning-Fog, Linsear-Write, and the Smog indexes).

However, all the correlations are still low, and we cannot rely solely on them to make predictions about the data.

3.5 Custom vocabulary for Dale-Chall formula

Dale-Chall's readability formula makes use of a list of simple words that was created in 1984 and revisited in 1995 based on the knowledge of fourth-grade students at that time. Technical texts on the Hyperskill platform are aimed at

	Flesch-Kincaid		**Gunning Fog**	
	score	p-value	score	p-value
norm. seconds to complete	0.167	0	0.123	0.00006
completion rate	-0.061	0.04502	-0.024	0.44312
avg like	0.021	0.48733	0.028	0.36409
topic completion rate	-0.017	0.5832	-0.04	0.19355
back to theory times per user session avg	0.128	0.00003	0.08	0.0088
back to theory users %	0.135	0.00001	0.077	0.01183
	Coleman-Liau		**ARI**	
	score	p-value	score	p-value
norm. seconds to complete	0.198	0	0.178	0
completion rate	-0.158	0	-0.075	0.01466
avg like	0.007	0.81527	0.016	0.59753
topic completion rate	-0.011	0.71649	-0.007	0.80938
back to theory times per user session avg	0.046	0.13514	0.132	0.00002
back to theory users %	0.083	0.00655	0.14	0
	Linsear-Write		**Spache**	
	score	p-value	score	p-value
norm. seconds to complete	0.129	0.00002	0.185	0
completion rate	-0.018	0.55232	-0.055	0.07373
avg like	0.021	0.48938	-0.011	0.72364
topic completion rate	-0.013	0.6632	0.002	0.94263
back to theory times per user session avg	0.127	0.00003	0.157	0
back to theory users %	0.12	0.00008	0.152	0
	Smog			
	score	p-value		
norm. seconds to complete	0.141	0.00066		
completion rate	-0.121	0.00362		
avg like	0.028	0.49847		
topic completion rate	-0.118	0.00436		
back to theory times per user session avg	0.101	0.01521		
back to theory users %	0.103	0.0134		

Table 3.3: Correlations of the Flesch and Dale-Chall scores with other readability metrics

readers whose vocabulary is larger than that of a fourth-grader, especially in more advanced topics that build upon previously explained material. I tried to extend the vocabulary of simple words to account for that, similarly to what was done in Gámez and Lesaux [2015].

As mentioned in section 2.1 where the main concepts of the platform are introduced, topics on Hyperskill are connected to each other via dependencies that are called *prerequisites*. Before reading a particular topic, students are obliged to finish all prerequisites of the topic. This structured nature of topics can be employed to create customized vocabularies. For each text, I created a separate vocabulary based on the texts of all its prerequisite topics, so that the vocabularies include words already familiar to the student.

Collecting information about topics' prerequisites. In the data dump with topic statistics that I was provided with and that I used for previously described tasks, for each theory step, there was only the information about its step ID. To extract a list of the topic's prerequisites, however, the corresponding topic ID was required. For this and the following operations, I made use of the Hyperskill API[3].

Firstly, given the theory step ID, I received information about its topic ID. Secondly, by a topic ID, I recursively collected all topic IDs of its prerequisites. The recursive approach is required since the list of a topic's prerequisites in API contains only direct prerequisites, while for my purposes, a full list of all prerequisites is required. As a result, I received a mapping from a topic ID to IDs of all topics that a user should have read before the current one. Finally, to receive texts of all prerequisite topics, using the topic IDs I extracted the information about topics' theory steps IDs.

Collecting the vocabularies. As a way of collecting "simple" words from prerequisite topics, I decided to take top-N words that are either the most frequent or have the highest TF-IDF values. The first approach is reasonable if we assume that words that occur in the topic the most are also the ones that the student will remember the best. The second method is based on the assumption that the student will better remember the words that are the most characteristic of the topic.

To collect the vocabularies, I tokenized and lemmatized the texts using spacy[4] (Montani et al. [2022]), and calculated word frequencies and words' TF-IDF values. As a result, for each topic, I received its individual vocabulary collected from all its prerequisites. For topics without prerequisites, it was the initial list by Dale-Chall, and for all other topics, it was a set of words combining the initial Dale-Chall list and top N words (by frequency or TF-IDF) from every prerequisite topic.

Finally, I computed new Dale-Chall scores, with the expansion of the lists of easy words either through the most frequent words in prerequisites or the ones with the highest TF-IDF. This time, I implemented the score calculations manually because the previously used library did not allow for a custom vocabulary.

Finding the optimal number of words. The idea to modify the standard Dale-Chall score was supposed to lead to a better correlation between the new score values and the statistical characteristics of a text. The user statistics corre-

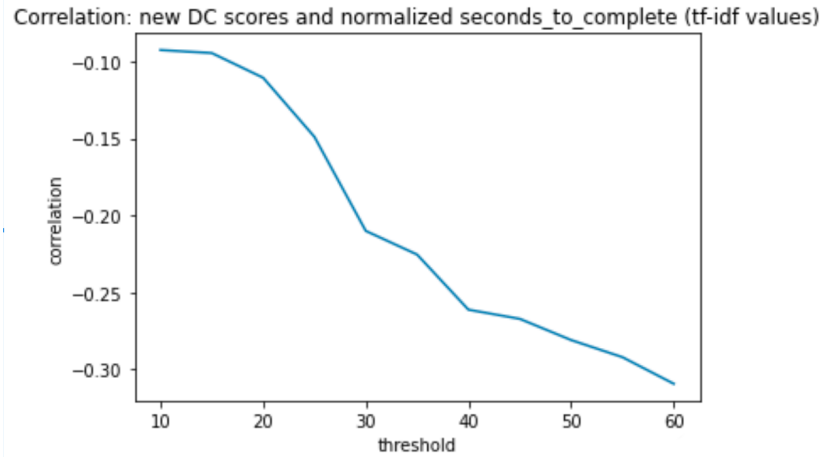
[3]https://hyperskill.org/api/docs/

[4]https://spacy.io/

sponding the most to the readability is the normalized seconds to complete, thus, to make a choice about the number of words collected from each prerequisite topic, I studied the correlation between it and the new scores.

For each N in the range from 10 to 60 with the step of 5, I calculated the new scores taking N words from each prerequisite topic to expand the vocabulary. Figure 3.3 shows the results of the correlation between the new scores and the normalized reading time for the TF-IDF version, with an increasing threshold for the number of words. We can see that with the larger thresholds, the correlation becomes more pronounced. However, it is negative, which implies that the more words we take from each prerequisite, the "easier" it is by the new score.

Figure 3.3: Correlation of the new DC scores depending on the threshold on the number of words



This result is unexpected but can be explained: the more words the formula considers "easy", the lower the resultant readability score is. Thus, if we take 60 words from each prerequisite topic, and if there are a lot of them, the score will be lower, whereas in reality, reading such a topic would require more time. This is confirmed by the fact that the correlation between the new score values when N=60 and the *number* of prerequisites was considerably high: -0.6. Once again, it explains the dependency: the more prerequisites a topic has → the more various words occur in the top 60 words from each prerequisite → the smaller number of words in a text is considered difficult by the formula → the easier the text is predicted to be.

As a result, the enhanced Dale-Chall formula led to smaller scores for our texts in general, which was part of our goal (we did not want most of the texts to be attributed to college students and graduates). However, the new scores were not better than the initial ones for predicting statistical metrics of texts, thus, there is no point in using the new formula instead of the original one, and we need to find more relevant metrics for identifying readability.

3.6 User-wise seconds to complete

Seeing that the parameter correlating with the readability scores the most consistently proves to be the normalized seconds to complete, I decided to investigate this metric in more detail.

In the data dump with statistics that I used, seconds to complete is given as the median value of individual users' completion times. The new idea was to see how individual users' completion times are distributed in comparison with the median values, and what their individual correlations with scores are (as opposed to the correlation of median topic values, as before).

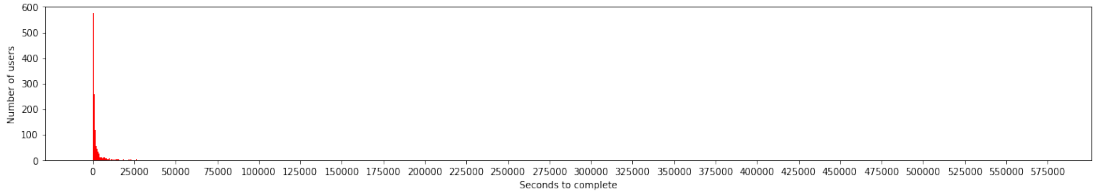
Even though the individual's reading time is influenced by their knowledge level, reading ability, level of English, and other factors that we cannot take into account, I wanted to confirm whether the median value was a reliable attribute to characterize all users.

Data description. The data I used for this task was in the form of a table where rows corresponded to anonymized users' IDs and columns – to topics. In each intersection, there was the parameter seconds to complete for a particular topic and a particular user (or Nan, if this user hasn't completed the topic). I call this a *user-topic dataframe*. For each topic, I also had the median seconds to complete from the previous statistics dump that I call the *topic statistics dataframe*.

Cleaning. The initial user-topic dataframe required some cleaning. Topics that were not in the topic statistics dataframe were dropped: they were either the new ones or those that didn't have enough completions and I had deleted them from the topic statistics dump. The resulting dataframe is used for topics' distributions in the rest of this section. The next step of cleaning was from the side of the users: only those rows (users) who completed at least 20 topics were left, otherwise, the correlation on the smaller number of instances wouldn't make sense. This version is used when computing correlations for all individual users.

Users' seconds to complete distribution for one topic. A random topic (ID 1933) was selected and I plotted the individual users' seconds to complete to investigate how this parameter changes. The results showed that the distribution was very dispersed (rf. figure 3.4).

Figure 3.4: Distribution of the individual users' seconds to complete for one topic

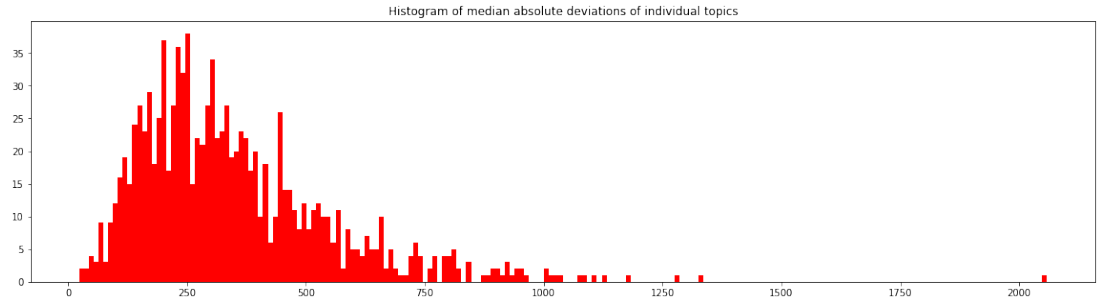


It can be seen from the graph that the majority of the values are close to 0 (but it is not 0 in reality, it seems so due to the scale), and there are some outliers with very large completion times. The statistical characteristics for this distribution were as follows: the standard deviation was 24302, the mean was 3531, and the median value was 613.

Such high results of standard deviation are explainable by the fact that due to some outliers, the mean value for the topic is higher than the majority of the values on the one hand and much lower than the outliers on the other. This is also the case for other topics. The median value, on the contrary, is not affected by outliers. Thus, to better understand the distributions, I decided to study the relations of values with their median.

Median absolute deviation (MAD). For each topic, MAD was calculated and plotted (figure 3.5).

Figure 3.5: Mean absolute deviation across individual users' completion times, for all topics



These results show that for the majority of topics, on average, the users' completion times differ from the median value by not more than 750 seconds (12.5 minutes).

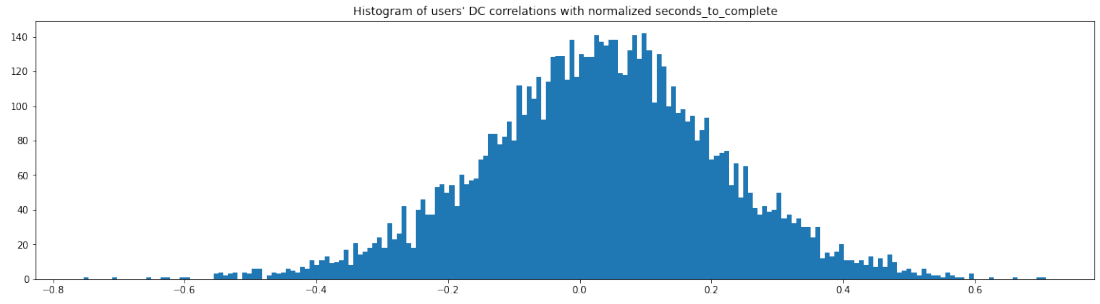
Based on this, I make a conclusion that even though the individual completion times for topics have large dispersion and a considerable number of outliers, the median value characterizes the data well and can be used as a good indicator of the time it takes to read the corresponding text.

Individual users' correlations with the scores. Finally, I also wanted to analyze correlation coefficients between the Dale-Chall and Flesch scores of topics and completion times of individual users. The previously computed correlations between the seconds to complete parameter and the readability scores took into account only the median value of completion time for each text instead of individual users' times. Now, for each user who completed more than 20 topics (there were 7760 of them), I calculated the Spearman rank correlation between his or her completion times for theoretical steps and the corresponding readability scores of those texts. The motivation behind this was to see if there were users whose completion time correlated highly or poorly with the scores and to assess the overall distribution of individual correlations.

The distribution of values for both Flesch and Dale-Chall scores turned out to be similar to the normal distribution, centered around 0. The example for the Dale-Chall scores is illustrated in figure 3.6; the results for the Flesch reading ease score are very similar. There are almost as many students with a positive correlation between their individual topic completion times and the readability scores, as there are with the negative one. This result emphasizes that topic completion times largely depend on the user and their individual factors more than on the readability scores as measured by the examined formulas.

It is worth mentioning, besides, that because of the multiple comparisons problem we cannot fully rely on these graphs featuring such a large number of correlation results. The graph shows the results of correlation tests of several thousands of users, and even if each particular score has an acceptable significance level, when we look at their aggregated result, the overall significance level is not under 0.05 since the probabilities of errors accumulate. However, the main conclusion still holds: all users complete topics at a different pace, and there are

Figure 3.6: Correlations between individual users' completion times for topics and the Dale-Chall scores for the topics



students whose completion times correlate positively and to quite a large extent, as well as those with a negative correlation, or absence of any.

3.7 Summary

In this chapter, I studied the data, investigated the basic readability scores and their correlations with statistical metrics of texts, and experimented with readjusting the Dale-Chall score for Hyperskill texts.

Some important results from this section are as follows:

- When analyzing correlations between statistical metrics of texts themselves, it was found that two metrics, the average like and the topic completion rate, do not have any correlations with any other metrics. However, for the rest of the metrics, seconds to complete and back to theory users % showed some correlations (between 0.43 and 0.97).
- Customizing the Dale-Chall score by expanding the list of easy words did not lead to better results: the new values correlated to the number of prerequisites a topic had, and the formula considered topics with a larger number of prerequisites to be easier in general.
- Completion times for individual users vary greatly, and there are some outliers with a very large completion time. However, the median value captures the times overall rather well.
- For individual users, their completion times for various topics may correlate with the readability scores either positively, negatively, or not have any correlation at all. There's no pattern seen in it.

In general, the results show that the basic readability scores, even enhanced for our texts, cannot be used to predict any of the statistical metrics of the texts. The fact that the correlations were so low implies that associations in the data are more complex than can be captured by shallow text parameters in the classical readability formulas. Even though some correlations could be seen, they were so weak that we couldn't ground any judgments about texts' readability on them or transfer them into any practical consequences. This result offers possibilities

to explore new ways of measuring readability. This is why the next chapter is devoted to machine learning techniques that are based on more complex text characteristics.

4. Experiments with machine learning

The task of identifying the readability of texts would be easier if there were gold evaluations of how readable the texts I work with are. However, automatically assessing readability is complicated for two reasons. First of all, readability itself is not an easily definable concept. It is connected to text comprehension, for which there is no simple way to measure it. As Dubay [2004] mentions, what results of reading tests reveal is not necessarily comprehension but, possibly, other artifacts such as prior knowledge, memory, or the difficulty of the questions. Besides, I work on a real-world task, and there is no estimation of readability for the texts I am dealing with. This is the specificity of my work that makes it both challenging and more exciting.

In the previous chapter, I considered statistical metrics of a text to be connected to readability, especially the normalized time it takes a student to read 10 tokens of a topic. I worked with the classical readability metrics and tried to find correlations between them and the statistical metrics. The results showed that the standard readability metrics were too weak to capture readability.

In this chapter, I describe my experiments with supervised machine learning algorithms. I formulate two tasks that I solved: regression predicting some of the statistical metrics, and binary classification for distinguishing between "good" and "bad" texts. I review the features that I used and specify how I collected them. Next, I explain how I divided the texts into "good" and "bad" ones based on comments that the users leave under them. Finally, I mention how I trained the models, analyze the results, and suggest a general procedure to assess the readability of the existing and new texts.

4.1 Task formulation

4.1.1 Regression for statistical metrics

Following the ideas of the previous chapter, one direction that I explore in this one is a prediction of the statistical metrics of a text, based on some of its characteristics. Namely, I predict three parameters:

- Average like of the text. It is averaged from the individual users' evaluations of the step on the scale with 5 emojis, from the angry face (corresponding to -2 on the scale) to a happy one (corresponding to 2).
- Completion rate of the theory step. It corresponds to the ratio of users who completed the step to those who opened it, and therefore ranges from 0 to 1. This idea is similar to those from works on user performance prediction (Baashar et al. [2021]).
- Seconds to complete, normalized by 10-token chunks. The median value across the 200 most recent completions is taken. This metric corresponds to

the reading speed of a particular text, which was recognized as an important parameter for readability.

For already existing topics, these metrics are available, meaning that the target features, in this case, are given. If the models work well, we will be able to estimate these parameters for a new topic and decide whether it is ready to be published or requires additional improvements.

4.1.2 Collection of readability assessments from users

Although average like, completion rate, and normalized completion time are connected to various readability aspects, they are not the same as readability. To receive readability assessments for topics, the first idea was to collect them from the users. Following a discussion with Hyperskill, they agreed to introduce a new evaluation scale for texts instead of the one collecting likes from the students. In the new evaluation scale, users would be asked how easy to read a particular text was.

Before implementing the new evaluation scale, however, it was required to make a prediction of whether the data collected in two months would be enough for me to use and make conclusions based on it. An important restriction was that the feature would be added for beta users (the ones who chose to see the "beta content" on the site) instead of all students.

To make the prediction, several steps were implemented:

- I approximated how many evaluations a content unit should have, for the average score to be consistent enough. This was mentioned before, in section 2.4 while describing statistical metrics available for texts: the result was that 20 reactions are needed for the average and standard deviation of likes to stabilize.

- I estimated the number of units required for the correlation computed on them to be significant. To approximate this number, I used the algorithm for sample size estimation as implemented on the site of the Chinese University of Hong Kong[1]. For the correlation coefficient r=0.4, approximately 50 topics are enough; for r=0.3, it should be 85 topics or more.

- Finally, I needed to confirm that the required number of topics estimated in the previous step is likely to get 20 or more reactions, with the evaluation scale seen by beta users for two months. To do that, I was given data about beta users' activity in the last two months. However, I found that only 27 topics received >= 10 evaluations from beta users in the last 2 months, which is not enough for my purposes (while for non-beta users, there are 434 topics with >=20 evaluations in the last month).

Thus, it made no sense to introduce the new evaluation only for beta users: the needed amount of data would not be collected.

[1] https://www2.ccrb.cuhk.edu.hk/stat/other/correlation.htm

4.1.3 Classification based on comments

Since it was not possible to get readability assessments from users, a new method of splitting texts according to their readability was required. There was no predefined information about readability, and it would have been difficult to receive a fine-graded scale of how easily a text was understood. Therefore, I decided to turn this task into a binary classification one: to divide all texts into two groups – well- and poorly written ones.

On Hyperskill, content teams that improve the texts on the site base their judgment not only on statistical metrics of a content unit but also on comments and feedback that students leave. In my work, I also decided to take comments into account to split the topics into the ones that require improvements and those that do not. I implemented and evaluated different approaches for that, and chose the one that received the best results. The details of this procedure are described in the following sections.

For the already existing topics, the company will be able to use this division to further improve the topics in the "bad" group, and this will be one of the results of my work. However, a different method is needed to classify new texts that do not have any comments. For that, I trained binary classification models, using the division of topics received in the previous step as the training corpus. The features I used for these models were almost the same as the ones for the regression task, and I describe all of them in the following section.

4.2 Features

In the regression and classification models, I used almost the same set of features, belonging to five different types:

- Linguistic features include various stylometric parameters of a text; with them, I try to capture the *style* of the topic;
- Meta-features refer to some characteristics of a topic *structure*, as well as implicitly model *prior knowledge* of the audience and *concept difficulty* of the text;
- Statistical features are the same user statistics used in chapter 3 to trace correlations; I consider them to be a proxy of readability;
- LM probability models how probable the text of the topic is, as measured by the GPT-2 model;
- LSA features are the result of latent semantic analysis on the texts, with 10 components corresponding to the weights of 10 main features in the document; implicitly, they can also reflect the *conceptual difficulty* of a text.

In this section, I describe all of them in detail, as well as the ways of their collection.

4.2.1 Linguistic features

Linguistic features were selected to represent the stylometric characteristics of a text. Among the available tools, Quita Up[2] was chosen (Kubát et al. [2014]) since it implements a wide variety of such features. What is more, it provides an R library[3] that is convenient to use.

1. **H-point**: a place in the frequency distribution of a text where the rank of a given word equals its frequency. It can be seen as an approximate threshold between function words and content words: all words above the h-point are supposed to be function ones, whereas the words below are content ones. It is also used to compute the Thematic Concentration of the text (see below).

2. **Entropy (H)**: defines the degree of vocabulary diversity – the greater the value, the more diversified the vocabulary is.

$$H = \log_2 N - \frac{1}{N} \sum_{r=1}^{V} f_r \log_2 f_r \tag{4.1}$$

3. **Verb distance**: arithmetic mean of the number of intervening tokens between two consecutive verbs in a text.

4. **Activity**: the degree of "action" of a text, in contrast to "description"; the ratio of verbs to the sum of verbs and adjectives in a text.

5. **Average token length**: the arithmetic mean of the token lengths in a text, in characters.

6. **Thematic concentration**: how much the text focuses on the main topic. The main topic is considered to include thematic words, which in their turn are all content words occurring above the h-point. Due to the nature of the h-point, the occurrence of the content words in the area above the h-point is considered a certain anomaly caused by the importance of the word in a given text. The resulting value of the thematic concentration is given by the sum of the so-called thematic weights of individual thematic words.

$$TC = \sum_{r'=1}^{T} 2\frac{(h - r')f(r')}{h(h-1)f(1)} \tag{4.2}$$

7. **Moving Average TTR**: an index of lexical richness. It is based on the calculation of Type-token ratio (TTR, Lively and Pressey [1923]) for text windows and taking the average for them. As the name suggests, TTR is the ratio of the number of types (unique words) to the number of tokens in the text. For the moving average, I used the window of 50 tokens because it roughly corresponds to two sentences (there are 743 tokens in a topic on average, and 32 sentences, so each sentence is around 23 tokens). I also tried the windows of 25 and 100, but in both cases, values from the 50-token window had a high Spearman rank correlation with the other window values, which is why I decided to leave only the 50-token window.

[2]https://korpus.cz/quitaup/

[3]https://github.com/czcorpus/QuitaUp

8. **Dale-Chall score**: the score that I also used in chapter 3 in the experiments with correlations. Even though the score itself was not capable of predicting statistical metrics, it might be useful as a part of the model.

9. **Flesch Reading Ease score**: similarly to the Dale-Chall score, I add it as a feature for machine learning models.

10. **Average sentence length**: the parameter that was found to be one of the most important criteria for text readability Feng et al. [2010]. For this reason I add this parameter separately.

Some of these values depend on the text length: e.g. the longer the text, the larger the H-point and the lower the entropy. However, since all of the theoretical steps we work with are similar in terms of their length, I disregard these dependencies.

4.2.2 Meta features

Meta features reflect some characteristics of a text's structure and dependencies with other texts. I used six such features:

1. **The number of direct prerequisites of a topic**. As mentioned previously, each text on the platform is connected to other texts through prerequisites: it means that before reading a particular text, the student needs to complete all its prerequisites. Direct prerequisites are the "first-level" prerequisites, and this feature stores their number.

2. **The number of overall prerequisites**. Overall prerequisites include all prerequisites of prerequisites, up until topics that have no prerequisites themselves. This feature corresponds to how "advanced" a topic is, or the concept difficulty of a text, to some extent: topics covering more complex concepts would usually require more prerequisite knowledge. Besides, it also can be seen as modeling the prior knowledge of students.

3. **The number of sections**. In general, each topic has 4-5 sections, but there are exceptions, and this parameter can play a role in the models' decisions. This feature belongs to structural ones and, what is more, it implicitly reflects the topic length.

4. **The number of images**. Though not every topic has images, the intuition behind this feature was that the presence of an image may help students understand the theory and therefore increase the overall readability of a topic.

5. **The number of code snippets in the text**. Similarly to pictures, not every topic includes code snippets, but their presence can make understanding a text a bit more difficult, for example. This is another structural feature.

6. **The ratio of the code snippets' length to the text length**. What may matter for readability is not only the number of code snippets in a topic but also what part they take compared to the rest of the topic. This is why

I computed the ratio of the overall code snippets' length to the length of the text, in symbols.

The features dealing with the prerequisites were collected through the Hyperskill API, similarly to how it was done while creating customized vocabularies for the Dale-Chall score. All other features were collected when the text was extracted from the XML representation, as described in section 2.2.

4.2.3 Statistical features

Statistical features include those that are taken from the information about students' completion of the material on the site. They were described in detail in section 2.4 but here I briefly list them again:

- **normalized seconds to complete** – how many seconds it takes for a user to complete the theory step (median value across users), normalized by chunks of 10 tokens;
- **completion rate** – the ratio of users who completed the step to those who opened it, ranging from 0 to 1.
- **avg like** – average rating of the step, ranges from -2 to 2.
- topic completion rate – what part of the students who tried to complete any of the topic's stages (theory or tasks) actually completed the topic.
- **back to theory users %** – percent of users who got back to the theory while solving the tasks.

As can be noticed, four of the previously used statistical features are not taken here:

- back to theory times per user session avg – this parameter had a very high correlation with the second "back to theory" metric, and I decided to include only the second one, as the percent of the users seems a more meaningful metric than the number of times the users got back to the theory.
- completed step users count, completed topic users count, and likes count – they were used to filter the data but do not bring any characteristic information per se, that is why they are not taken as training features.

4.2.4 LM probability

Large language models, such as GPT-2 (Radford et al. [2019]), can be used to estimate how probable a text is. This information can be used to assess, for example, how natural a text sounds Zhang et al. [2021]. We can assume that a long and complex sentence will be estimated as less probable than a clear and concise one. Following this idea, I receive probabilities for all topics and use them as an additional feature for ML models.

To receive the probabilities, I used a wrapper around the GPT-2 model, `lm-scorer` (Primarosa [2020])[4]. I firstly split each text into sentences using spacy[5], then for each sentence I received the probabilities of its tokens and took their geometric mean to have one score for each sentence. Finally, I averaged the scores from all sentences in the text to receive one score per topic.

It is worth mentioning why I calculated the mean of token and sentence probabilities instead of multiplying them, which is sometimes used in such settings. There are two main problems with multiplication. The first is that its result is influenced by the length of a text: sentences with more words will receive a much lower probability estimation than those with fewer words. Secondly, when multiplying already low probability numbers, I would receive numbers very close to zero, which would be difficult to operate with. This is why I made a decision to average the results of the estimations.

4.2.5 LSA features

Latent Semantic Analysis (Deerwester et al. [1990]) is a topic modeling technique based on Singular Value Decomposition, a method for dimensionality reduction. In LSA, a document is considered to be generated by a mixture of topics; each topic, in turn, is represented by words together with the numbers specifying how likely each word is to occur in this topic.

Essentially, LSA provides vector representations of texts, which can later be used to extend a feature set of a document. Every value in the vector represents how important a specific topic is for the document. This has been proved to work for classification tasks (Daneshvar and Inkpen [2018], Zhang et al. [2019]), so I decided to add these features to my dataset.

Prior to the decomposition, I lemmatized the texts, deleted stopwords, and represented them as TF-IDF vectors. I used the SVD implementation from the `sklearn` library (Pedregosa et al. [2011]) and decided to take the first 10 components: a larger number of features would increase the feature set too much. The generated topics are shown below:

1. Topic 1: **general concepts** ['math', 'method', 'function', 'class', 'element', 'value', 'string']
2. Topic 2: **mathematics** ['math', 'matrix', 'vector', 'node', 'graph', 'linear', 'probability']
3. Topic 3: **OOP** ['class', 'method', 'object', 'instance', 'constructor', 'field', 'math']
4. Topic 4: **collections** ['array', 'element', 'index', 'list', 'method', 'collection', 'value']
5. Topic 5: **functions** ['function', 'string', 'variable', 'character', 'argument', 'value', 'operator']

[4]https://github.com/simonepri/lm-scorer
[5]https://spacy.io/

6. Topic 6: **strings** ['string','character','match','pattern','substring','method', 'regex']
7. Topic 7: **?** ['table', 'thread', 'loop', 'operator', 'query', 'statement', 'column']
8. Topic 8: **program execution** ['thread', 'loop', 'exception', 'program', 'array', 'method', 'execute']
9. Topic 9: **files and OS** ['file', 'loop', 'array', 'operator', 'variable', 'type', 'directory']
10. Topic 10: **databases** ['array', 'thread', 'table', 'column', 'database', 'query', 'row']

As we can see, almost all of them, except for the 7th topic, can be easily interpreted. The weights of the topics in a particular text can also reflect its *conceptual difficulty* to some extent: some topics may be inherently more complex than others. How well these features contributed to the performance of the models is analyzed in the following sections.

4.3 Corpus of good and bad topics based on comments

In this section, I describe the algorithm for dividing the topics into those that would need some improvement and those that are well written. Users often share their opinion if they found the text to be hard to read. Examples of such comments are provided in 4.1.

Comment
I have trouble understanding this topic. Is it really that complicated or just badly worded? Also, the examples are too heavy for a beginner, maybe you can explain each keyword with few but simpler code snippets?
It was the hardest theory so far. Not because the topic is hard, but because it's really bad explained for students with zero HTML/CSS experience.
The wording is weird here. The phrase "reference to the same value" threw me off a little.
Language is unclear. Consider if it's the only way it can be done or not.

Table 4.1: Examples of negative comments

My goal was to identify such comments and then make a decision about a topic's readability based on them. The division of topics in such a way was my approach to measuring text comprehension, an aspect that is always considered to play a large role in readability. The process consisted of several steps.

Firstly, I analyzed 50 of such comments manually and created a vocabulary of verbs, nouns, adjectives, and adverbs that the students used to express their dissatisfaction with the wording, language, or overall comprehensibility of the texts.

Secondly, I expanded this vocabulary with the synonyms taken from a thesaurus available online[6]. For each word in the initial list, I manually checked its synonyms and added those that were semantically relevant for the sense in which the word was used in the initial comment. These form the list of what I call "suspicious" words that you can find in A.1. It is worth mentioning, however, that among these words there are some that do not have any positive or negative connotation, e.g. "description", "lesson", or "comprehension". What is more, some of the words, especially adjectives, could refer to positive characteristics of a topic, e.g. "evident", "readable", or "precise", while in the negative comments, they could be used with a negation. Therefore, comments that contain such words may be either positive or negative, and an additional step of filtering was required. I tried several ways of further filtering and division of topics according to the comments. In the rest of the section, I describe these methods. It is important to explain this procedure in detail because the results of these topic divisions are used to distinguish between good and bad topics among those published on the site, as well as serve as a training corpus for classification described in the following sections.

4.3.1 Version 1

After receiving a list of "suspicious" words in the previous step, I first counted the number of such words in each comment. Comments that do not contain any "trigger" words were automatically classified as neutral ones. On the comments with suspicious words, I performed sentiment analysis to identify which of them are positive and which are negative.

For sentiment analysis, I used two models: `textblob` Loria [2018][7] and `flair` Akbik et al. [2019][8]. Having calculated two scores from the models, I split the comments into "good", "bad", and "vague":

- The ones that received two positive scores from the sentiment models are "good";
- The ones that received two negative scores are "bad";
- If the scores are close to zero or the polarity differs between the two, such comments are "vague".

The distribution of comments classified as positive, negative, or "vague", can be found in A.2, for this and the following versions of the algorithm.

As a polarity threshold, I took 0.25 for `textblob` and 0.5 for `flair`, based on the distribution of polarities among the comments. The polarity threshold, in this case, defines the minimum score for the comment to be classified as a positive one; otherwise, it is considered neutral. The negative threshold is symmetric.

[6]https://www.thesaurus.com/
[7]https://textblob.readthedocs.io/en/dev/
[8]https://github.com/flairNLP/flair

Next, based on the division of comments into good and bad ones, I split the topics:

- The topics that have 1 or more positive comments and no negative ones are considered good;
- The topics that have no positive comments and 1 or more negative comments that are not "fixed" are marked as bad. "Fixed" is a status that a comment may have that means that it has been analyzed by the content team, and a topic has been changed accordingly – therefore, such comments should not be counted as bad ones because they are outdated;
- If topics contained no comments with "suspicious words", they are also "good";
- All topics that contained both good and bad comments are not attributed to either group.

Finally, I further filtered this division by comparing the "bad" and "good" groups to the list of topics with a very high average like of >1.85 (out of 2). This step was due to the assumption that the topics with a high like should belong to the "good" group and cannot have problems with readability. However, there were some intersections with the "bad" group, so I deleted the theory steps with the high rating from the "bad" group and expanded the "good" group with them.

This method classifies the topics into strictly "good" and strictly "bad" ones (according to the sentiment scores of their comments), neglecting the ones in between. The main disadvantage of such an approach is that a lot of topics cannot be attributed to either "bad" or "good" groups: out of 1120 topics that have comments, only 467 are left, with 132 (28%) "bad" ones and 335 (72%) "good" ones.

4.3.2 Version 2

In the previous version, two sentiment analysis libraries were used, and comments were classified as positive or negative only if the scores from both libraries matched. However, it excluded a large number of comments that could not be classified as either positive or negative. To partly tackle this problem, in this version I added another sentiment analysis library, `vader`[9] Hutto and Gilbert [2014], which was created specifically for social media texts and could therefore introduce a valuable additional evaluation. Having three scores for each comment, I divided the comments into positive and negative ones as follows:

- If the majority of the scores were positive, the comment is considered positive;
- Vice versa for negative comments;
- In other cases: the comment can't be classified and is considered vague.

[9]https://github.com/cjhutto/vaderSentiment

Apart from having three scores, I also increased the polarity thresholds for the models: for this version, they were 0.5 for `textblob` (instead of 0.25), 0.75 for `flair` (instead of 0.5), and for the `vader` model it was 0.5. This was done to make the predictions of the models more reliable.

In all other aspects, the version was similar to the first one and therefore had the same disadvantage: out of 1120 topics that have comments, 487 were left (this is by 20 more than for the first version), with 126 (26%) "bad" ones and 351 (74%) "good" ones.

4.3.3 Version 3

The next version featured several improvements. First of all, I excluded adjectives and adverbs from the list of the "suspicious" words, since they usually bear a significant sentiment load. The corresponding comments would be classified as good or bad and would be counted in the topic evaluation, even if these adjectives referred to some other subject than text's comprehension. Therefore, the list of comments that are analyzed in this method is now shorter and includes only nouns and verbs.

The sentiment analysis results are then taken for comments, but they are classified not into positive and negative ones but rather into negative and the rest, by the majority vote of the sentiment models, as before. If at least two of the three models evaluated a comment as a negative one, it is marked accordingly; otherwise, the comment is considered neutral.

The procedure of splitting the topics based on comments was also changed: now, topics that have >=3 negative comments are considered bad ones, whereas everything else is considered good, or rather, "not bad" topics. This way we split the texts not into "bad" and "good" but into "bad" and "everything else". It allows us to use all available topics and is intuitively more correct since, for a human, it is easier to identify a poorly-written text than to discriminate between an acceptably- and a well-written text. A text with bad readability would mean that it is difficult to understand the content, which is not hard to notice, whereas normal and high readability levels both provide acceptable comprehensibility, and the distinction between them is neater.

Finally, in this version, I also do not extend the group of "good" texts with all highly-ranked topics, and, similarly, do not exclude the highly-ranked topics that ended up in the "bad" group. This decision was made after I learned that even with such a strict requirement on three or more negative comments for a topic in order to consider it bad, some highly-ranked topics still ended up in the "bad" group. Seeing this, I analyzed the comments for such topics to understand whether they are classified correctly. I learned that even though they received high evaluations from the students, the comments did contain some complaints and requests for improvement. An example could be a topic on recursion in Python which had an average like of 1.91 but was attributed to the "bad" ones by the algorithm and contained the following comments among others:

- *The example is hard to understand.*
- *Honestly, I think everyone would be better off if they just skipped reading this one.*

- *I can't believe how well-written this piece is! And I'm not joking, like seriously!*

This, indeed, is a difficult case but it proves that a high average like does not guarantee that everyone understands the topic well. I, therefore, decided not to alter the good and bad groups classified by the algorithm because of intersections with highly-ranked topics.

This version yielded all 1120 topics, with only 89 (8%) bad ones and 1031 (92%) good or rather "not bad" topics.

4.3.4 Version 4

In the previous version, I didn't use the whole initial list of "suspicious" words anymore, because some comments that included these words did not really refer to readability but to some other subjects. In this version, I followed a similar procedure but instead of taking nouns and verbs, I manually selected words that refer to writing, reading, and understanding. This would allow me to limit the search only to comments that mention something about these topics. The rest of the procedure was very similar to the one in the previous version:

- Count suspicious words in comments;
- Look at the results of the sentiment analysis;
- Mark comments as negative if they are classified as such at least by two of the three models;
- Take bad topics as those with at least 2 negative comments

This version produced 142 (13%) bad topics and 978 (87%) good ones.

4.3.5 Version 5

In the previous versions, I saw that the sentiment analysis assessments of the comments did not always work well, even with the majority vote and higher thresholds for models. Some comments that were classified as negative ones, in reality, were not. To improve this, I decided to skip the sentiment analysis step but choose the negative comments based solely on their vocabulary. I created two lists from the "suspicious" words:

- one with words about writing, reading, and understanding (the same as in the previous version),
- and a second one with words connected to something being unclear or difficult.

Next, I searched for comments that had at least two words, one from each of these lists. These comments were marked as negative ones, and all others were considered neutral. Division of topics based on the comments was performed as before: bad topics were those that had at least 3 such "bad" comments.

This version returned 223 (20%) bad topics and 897 (80%) good ones.

4.3.6 Version 5 with likes

Finally, there was another piece of information in the dump of comments that could be used to improve the topics' division: the number of positive reactions a comment received.

This version was almost the same as the previous one except for one detail: when computing the number of negative comments a topic had, they were counted additionally as many times as there were likes, based on the assumption that if a student put a positive reaction for a negative comment, they agree with it, and so the comment could have been duplicated. This approach resulted in more negative topics: 317 (28%) as opposed to 803 (72%) positive ones.

The short overview of all approaches is shown in table 4.2.

4.4 Evaluation of the corpus

After creating the divisions of topics into good and bad, an evaluation of the results was required. There was no gold division of topics, so I performed a manual evaluation. Assessing each topic of every version would be an impossible amount of work, so I followed another approach:

- From each system's division, I randomly chose 10 good and 10 bad topics.
- I combined these topics from all the systems into one evaluation form for two independently working annotators while hiding the version names and scores from them.
- The annotation was carried out by analyzing the comments for every topic: if the comments contained complaints about something being unclear, the topic was to be marked as bad; if there were no such comments or only few ones compared to many positive ones, such a topic was attributed to good ones.
- Finally, for each version, I calculated the number of true and false positives, as well as true and false negatives, and the F-score for each of the versions.

There were two annotators: me and a person from Hyperskill who was familiar with the process of reviewing user feedback and so could evaluate the quality of a topic from a more professional point of view. The criteria for attributing a text to the "bad" readability group were formulated as follows: the comments should contain complaints about some aspects of the text not being clear or requests to explain something in more detail or in a different way. However, comments that suggested adding some additional information except for the already present in the text were supposed to be disregarded. Such comments do not testify that the text was poorly-written but rather that a person would like to learn more about a topic.

The inter-annotator agreement between me and the second annotator according to Cohen's Kappa (Cohen [1960]) was 0.91, which is a very high value. Indeed, it is often evident from the comments whether some parts of the topic were not clear to the audience because students express their opinion on that. Interestingly, all cases when our annotations did not coincide were when I considered a

Ver.	Short overview	Topic grades	#bad	# neutral	# overall
1	Manual list of suspicious words; classify such comments with 2 sentiment models and take scores where they agree	Bad topics with >=1 negative comments and no positive ones + extend with highly ranked topics	132	335	467
2	The same as 1 but the majority vote of 3 sentiment models	The same as version 1	126	351	487
3	Suspicious comments - with nouns and verbs from the initial list, then the majority vote from 3 sentiment models	Bad topics with >=3 negative comments	89	1031	1120
4	Suspicious comments - with words about 'understanding', then the majority vote from 3 sentiment models	Bad topics with >=2 negative comments	142	978	1120
5	No sentiment analysis, "negative" comments contain words from two lists: "understanding" and "difficulty"	Bad topics with >=3 negative comments	223	897	1120
5 with likes	The same as 5 but count the number of likes for negative comments	The same as version 5	317	803	1120

Table 4.2: Summary of the approaches in all versions

text to be normal and the Hyperskill annotator marked it as bad, and most of these occurred in the evaluation of version 4.

Since we had a very large agreement on the scores of topics, and I consider the second annotator to be more qualified for this task, for the evaluation of models I use their evaluations. The results can be seen in 4.3.

Ver.	**TP**	**TN**	**FP**	**FN**	**F-score**	**Recall**
1	8	8	2	2	0.8	**0.8**
2	8	9	1	1	**0.9**	**0.9**
3	8	6	1	4	0.78	0.69
4	8	7	1	3	0.81	0.75
5	10	7	0	3	**0.86**	0.77
5 with likes	10	6	0	4	0.83	0.71

Table 4.3: Summary of the version evaluation

We can see that the best version according to both metrics was version 2, which introduced three sentiment analysis models and relied on the whole list of "suspicious words". This proves that using three models to assess the polarity of a comment is beneficial for the algorithms. However, for future work, I would like to use one of the versions 3-5 since they are capable of classifying all topics, and not only their part (versions 1 and 2 could classify only around 42% of the available topics, leaving the rest as "vague" ones). From versions 3-5, the best results are provided by version 5, which did not use sentiment analysis and only relied on the presence of certain words in the comments. Interestingly, the fifth version with the additional information about comments' likes did not outperform the original version 5. The worst results according to both metrics belonged to version 3 that limited the list of suspicious words by nouns and verbs.

As a result, the best values of metrics were achieved by the second version but it is not the one used in further experiments. For that, I selected version 5, as well as the second best algorithm according to recall, version 4. I decided to use both these versions, keeping in mind that the evaluation was performed on random topics taken from the algorithm's results, and small differences in their performance can be due to that.

It is also worth mentioning why I consider it appropriate to take 10 topics from good and bad groups **after** the results from an algorithm were received, rather than before, as would be done in a standard evaluation pipeline. There are several reasons for that:

- First of all, I wanted to receive a similar distribution of good and bad topics in the evaluation since only 20 topics from one version are taken. If I selected the topics before the work of the algorithm, I would have more "good" ones because their number is larger in general.

- Secondly, the results from all versions are combined into one evaluation file and shuffled, and what is more, the results themselves contained some errors, so the proportion of true good and true bad topics in the evaluation is not 1 to 1.

- Finally, the annotator from Hyperskill did not know how many models there were and how many topics from each of them were taken. As for my evaluation, even though I knew this, the information about from what version a particular topic was taken was not available to me at the time of the evaluation.

In the next section, I describe the models and results achieved by the machine learning algorithms, including classification built on the division of topics from this section.

4.5 Models and results

This section provides a detailed overview of models used for two types of tasks: regression predicting three statistical metrics and classification for identifying whether a text is understandably written.

4.5.1 Regression

This is a standard regression task, with the target features taken from the dataset with texts' statistics and training features including various sets of features described in section 4.2. I experimented with different types of features to analyze how important a particular type is for the overall result.

Out of 1321 original texts, I firstly filtered them by the number of completions, leaving only those steps that were finished by at least 20 students. This was required to make the statistical features more reliable (they are averaged across all or the last N students who completed the step). This resulted in 1061 topics left. Then, for the case with the average like, I further filtered the data by the same threshold of minimum 20 evaluations from users, which left 591 steps. From both of these corpora, 15% were left for testing, giving 901 and 502 instances for training, and 160 and 89 for testing, respectively.

Next, seven regression models were chosen from the ones available in `sklearn`: Linear Regression, Lasso, ElasticNet, SGD Regressor, Bayesian Ridge, SVR, and Kernel Ridge. For each of them, a grid with several parameters for tuning was made. Then, I performed a grid search for all of them, with the results evaluated in a 5-fold cross-validation, and saved the best parameters for each model. Finally, I evaluated the performance of every model and chose the best of them. The metric for evaluation was the mean average error (MAE) since it provides for better explainability. I repeated this procedure for several sets of features, and present the results in 4.4.

Average like. This is the experiment with the smallest training corpus of only 502 instances. The range of the average like in the training data is from 0.43 to 2.0, which means that the MAE of 0.16 corresponds to approximately 10% of the range.

On training data, the results of the best and worst models differ only by 0.0058, which is why it is difficult to make any conclusions about the features' importance. On the test data, however, the best results belong to the models that include the LSA features, and most interestingly, the one that includes only them is among the best as well.

	Average like			Completion rate		
Features	**Train**	**Test**	**Model**	**Train**	**Test**	**Model**
ling	0.1601	0.1714	Kernel Ridge	0.1299	0.139	Kernel Ridge
ling, meta	0.1588	0.1731	Kernel Ridge	0.1174	0.1189	Kernel Ridge
ling, LM	**0.1584**	0.1707	Linear Regr.	0.1295	0.1391	Kernel Ridge
ling,meta, LM	**0.1568**	0.1724	Linear Regr.	**0.1118**	0.1175	SVR
ling,meta, LM, LSA	0.1596	**0.1357**	Linear Regr.	**0.1143**	**0.1045**	Bayesian Ridge
meta, LM, LSA	0.1613	0.143	Linear Regr.	0.115	**0.1096**	Kernel Ridge
LSA	0.1626	**0.1419**	Linear Regr.	0.129	0.1168	Kernel Ridge
meta	0.162	0.1749	SGD	0.1261	0.1262	Kernel Ridge

	Normalized compl. time					
	Train	**Test**	**Model**			
ling	2.419	2.8888	Kernel Ridge			
ling, meta	1.9566	2.3846	Kernel Ridge			
ling, LM	2.3108	2.7747	Kernel Ridge			
ling,meta, LM	1.9397	2.3598	Kernel Ridge			
ling,meta, LM, LSA	**1.6887**	**2.1234**	SVR			
meta, LM, LSA	**1.7391**	**2.096**	SVR			
LSA	2.1865	2.5448	SVR			
meta	2.0754	2.45	Kernel Ridge			

Table 4.4: Mean Average Error for regression results

Completion rate. The range of completion rate in the training corpus is from 0.29 to 0.95, and the MAE of 0.11 corresponds to 16% of the range. Similarly to the average like, the results of the models on training data differ at most by 0.018, which is insignificant and does not allow to see which types of features contribute the most to the algorithms' performance. Another similarity to the results of the average like prediction is that the models that have the best results on the test data are those that include LSA as a feature. Also, by the results on the test data, we can see that adding meta features improves the generalization capacities of the models: MAE of "ling, meta" and "ling, meta, LM" are lower by approximately 0.02 than those of just "ling" and "ling, LM", correspondingly. However, as opposed to the case with LSA in average like prediction, meta-features by themselves are not providing the best results.

Normalized seconds. The range of normalized completion times in the training corpus is from 1.46 to 21.07, and the MAE of 2.0 corresponds to 10% of the range. Similarly to the predictions of completion rate, we can see that meta-features are important for the models: even by themselves, they give a reasonable MAE that is better than on LSA or linguistic features alone. When combined with other features, such as linguistic and LM, the results become better, reaching their maximum when LSA features are also added.

4.5.2 Classification

As explained in sections 4.3 and 4.4, to receive the distribution of topics into good and bad ones according to readability, I performed experiments with several rule-based algorithms and then evaluated their results manually. For the already existing topics, to capture those requiring attention, we can simply use the results of one of these algorithms. For new topics, however, another approach is needed, since they do not have comments to ground the search on. To solve this issue, I trained binary classification models, taking the division from the previous step as the training corpus. I chose algorithms from versions 4 and 5, since they showed good results and returned the division of all the topics rather than only a part of them, like versions 1 and 2 did.

With both version 4 and 5 the training process was the same, and similar to that of regression models described earlier. First, I split the data into train and test (85% and 15%, or 952 and 168 instances, respectively). Then I selected several classification algorithms implemented in `sklearn` (Logistic Regression, Ridge, K Nearest Neighbors, SVM, SGD, Passive Aggressive, MLP, Gaussian Process, Decision Tree, Naive Bayes), and built a grid for parameter optimization for each of them. After running a grid search with a 5-fold cross-validation on the training data, I retained the best model from each algorithm. Recall was chosen as the evaluation metric because of research priorities: it is a lesser mistake to erroneously classify a normal topic as a bad one than to miss a topic that needs improvement.

Finally, I ran an evaluation of their results on training and test data and kept the best model. I repeated this procedure for various sets of features. The results for version 4 are shown in the 4.5, and for version 5 in the 4.6.

Even though according to the manual evaluation, versions 4 and 5 were similar, we can see a drastic difference of performance of machine learning models on them.

Features	Recall		Model
	Train	**Test**	
ling	1	0.1429	Decision Tree
ling, meta	0.6281	0.2381	SVM
ling, LM	1	1	SGD
meta, LM	1	0.2857	Decision Tree
ling, meta, LM	0.686	0.2381	SVM
ling, meta, LM, LSA	1	0.4286	Nearest Neighbors
meta, LM, LSA	0.7603	0.2857	SVM
ling, meta, stat	0.6639	0.5238	Passive Aggressive Class.
stat	1	0.1905	Nearest Neighbors
ling, stat	0.3109	0.4286	SGD

Table 4.5: Classification results on the division of topics by Version 4

Features	Recall		F-score		Model
	Train	**Test**	**Train**	**Test**	
ling	0.3316	0.1818	**0.45**	0.24	MLP
ling, meta	0.8526	0.7576	0.3653	0.3185	Passive Aggressive Class.
ling, LM	**0.9158**	**0.8788**	0.3349	0.3152	Passive Aggressive Class.
meta, LM	0.3211	0.1515	0.3219	0.1563	Passive Aggressive Class.
ling, meta, LM	0.8316	0.7576	0.3624	0.3268	Passive Aggressive Class.
ling, meta, LM, LSA	**0.9053**	**0.8485**	0.3844	0.3636	Passive Aggressive Class.
meta, LM, LSA	0.8526	0.8182	0.3767	0.3776	SGD
ling, meta, stat	0.3883	0.4545	0.4022	**0.4286**	Naive Bayes
stat	0.2394	0.2424	0.2394	0.25	SVM
ling, stat	0.7287	0.4242	**0.8035**	**0.3944**	SVM

Table 4.6: Classification results on the division of topics by Version 5

In version 4, all models overfit on the training data, performing poorly on the test data. The only exception is the SGD classifier, but it simply returns the same label for all input data, making the model useless. Therefore, I don't analyze the results for version 4 and focus solely on version 5.

The division of topics in Version 5 is better captured by the models. However, despite impressive results of the recall metric, the overall performance of the models is poor, as illustrated by the F-score. The best F-score is obtained with the use of linguistic, meta, and statistical features, followed by the combination of linguistic and statistical features.

Statistical features are new compared to the regression task, where they were not used in training (some of them were the target feature for regression itself). Since this classification is aimed at new topics that could not be classified via their comments, the statistical features can't be used in the final system either: new topics don't have user statistics yet. However, it was a promising idea to see whether the topic's readability would be reflected by their statistics. We can see that statistical features reach low recall and f-score by themselves, but using them with other feature combinations helps the classification.

If we exclude statistical features, the best combination of recall and f-score are obtained by the two models that make use of LSA features: the one with "meta, LM, LSA", and "ling, meta, LM, LSA", which justifies the use of LSA topic weights as text features. However, LSA features could improve the performance only up to a certain point. This can be explained as well by the visualization of the texts from the two groups (figure 4.1): documents are not clustered in one or several areas, they are distributed across the whole area where good texts also occur. Practically, it also means that there are no inherently problematic texts for any of the 10 topics identified by LSA.

Figure 4.1: Visualization of the distribution of "good" and "bad" texts according to version 5

The visualization of the 10-dimensional document vectors was performed with the help of the Uniform Manifold Approximation and Projection for Dimension

Reduction (UMAP) embeddings McInnes et al. [2018].

Summing up, such performance is not suitable for a classification model that would be employed on Hyperskill. The features used for this task were not capable of distinguishing "bad" and "good" text according to the division received by the rule-based algorithm in the previous step.

4.6 General system

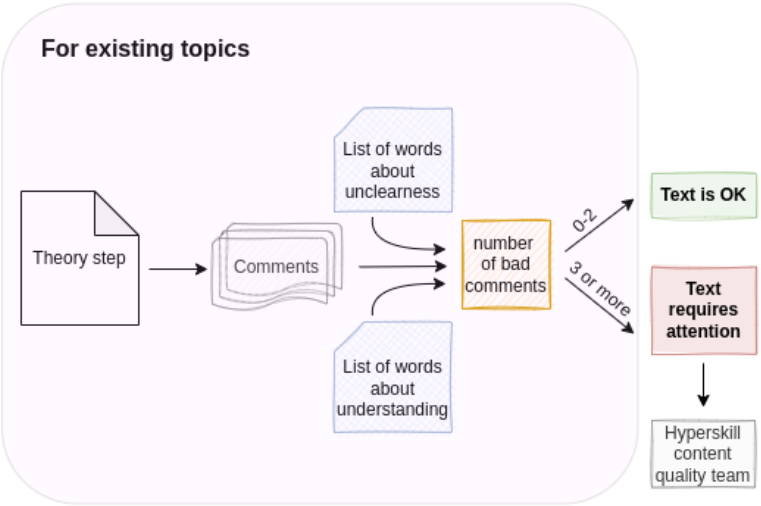
Having experimented with regression for the prediction of statistical metrics and classification for splitting texts into groups of well- and poorly-readable ones, I describe the general pipeline that I proposed for Hyperskill.

4.6.1 Topics that are published on the site

The process differs for topics that are already published on the site and the new ones.

For the already existing texts, the procedure is illustrated in figure 4.2. A rule-based algorithm based on their comments is used: version 5 that checks for the presence of certain words in the comments and marks a text as bad if it had 3 or more negative comments.

Figure 4.2: The overall pipeline for the existing topics



It is worth mentioning why version 5 is chosen for the final pipeline. The evaluation of the algorithms was carried out on a random subset of the classified data, and the best results were obtained by version 2. However, this version marked as "bad" topics only those that received no positive comments together with one or more negative ones. However, as we have seen, a text can have both positive and negative comments but require some improvements in terms of the language. It means that the second version would not be able to identify such topics as bad and is not suitable for us. The most important metric is recall, and we want as many bad topics as possible to be identified as such.

The use of the rule-based algorithm for identifying good and bad topics yields a list of texts that were identified as bad, and this is the result of my system for

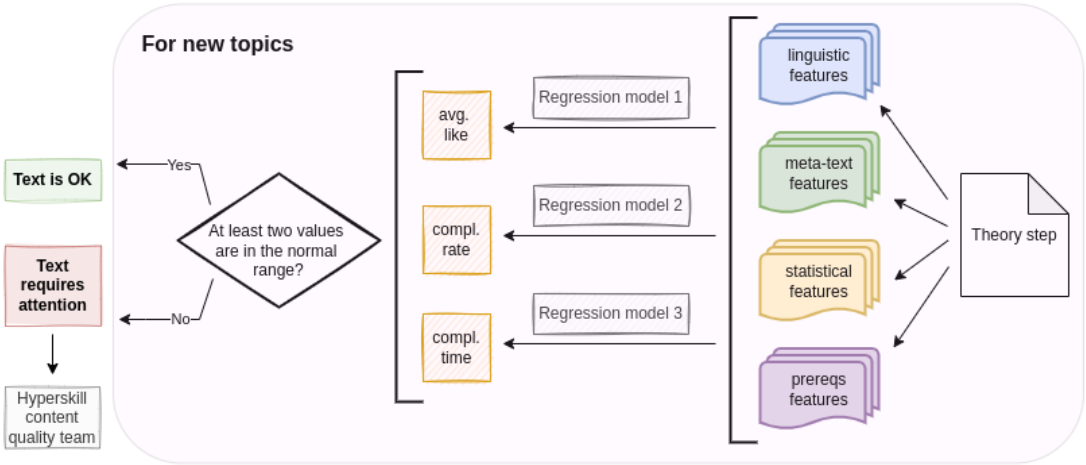
the already existing texts. This information can later be used by the company to improve these texts.

Even though classical algorithms usually have a scale of readability rating, a binary prediction does make sense. In the readability scales, a threshold is often specified depending on which texts are considered either suited for a specific audience or not. Thus, all metrics that map a text to its grade level presuppose that the text is well-readable for any grade higher than the mapped one (e.g. a second-grade text can be easily understood by fifth-graders). Binary prediction can be seen as a similar approach: it measures whether the text is understandable enough for an average user of the Hyperskill platform.

4.6.2 New topics

For the newly written topics, there are no user statistics and no comments, which requires us to use other approaches. The pipeline is shown in 4.3. The classification that was supposed to split them into bad and good groups after training on the division from the rule-based algorithm, did not show acceptable results.

Figure 4.3: The overall pipeline for the new topics



However, the results of predicting statistical metrics of texts were better, this is why the division of newly written topics is based on it.

First of all, from the textual theory step four kinds of features are extracted: linguistic, meta-features, LM-probability, and LSA ones. They are then passed to three regression models that predict the average like, completion rate, and the normalized completion time for the corresponding text.

On Hyperskill, the content quality team keeps a close watch on "suspicious" tasks and theory steps that are identified as such with the help of user statistics: average like and completion rate. For theoretical steps, the criteria are as follows: Avg_Like <0.8 OR Compl_Rate <0.4.

Following this idea, I create the list of "bad" topics from those whose predictions of user statistics is below the thresholds in at least two out of the three metrics. For the completion time, I analyzed the distribution of values from the training data and chose the threshold of 12.8, this being the 0.97 percentile of the

data. The intuition behind this approach is that such texts, even if not solely because of readability, deserve attention from the Hyperskill content team. Finally, just as with the results on the existing topics, the texts that were marked as bad are to be analyzed and improved by the experts from the Hyperskill team.

Individual parts of this pipeline were described and evaluated in the previous sections. However, the performance of the overall system is yet to be tested by the company.

4.7 Summary

In this chapter, I experimented with machine learning algorithms for identifying well- and poorly-written texts, since the research of the classical readability metrics carried out in the previous chapter proved that they are not capable of capturing text readability expressed in terms of user statistics. The two tasks I tried to solve with machine learning were regression for predicting user statistics of a text, and binary classification for distinguishing between "good" and "bad" texts.

The results of the regression task were satisfactory and could be used to predict user statistics for newly created texts in the overall system. The features used for the models included linguistic and meta-features, as well as LM probability and LSA topic weights.

For the classification task, I described the creation of the training corpus that was based on the comments for texts. I explained the experiments with various algorithms for discovering negative comments and "bad" topics. The results of the classification models were not satisfactory, therefore they are not used in the final system.

Finally, I introduced the overall pipeline for new and existing topics to identify those requiring improvements from the Hyperskill content team.

4.8 Future work

This section covers a few promising directions for future research.

4.8.1 Create a corpus based on the history of edits

For each textual unit on the website, a history of edits is stored in the platform's backend. In the course of my research, I planned to use it to automatically collect a corpus of changes that could be further analyzed or even used as training data for machine learning systems.

The idea was to collect texts that have recently undergone some changes, then leave only those where the changes affected the text itself (rather than a code snippet, for example), and where the changes were significant (as opposed to fixing a typo, for example). Thus, versions of texts before and after changes would have been gained, and statistical metrics of the corresponding unit could have been traced to see if the statistics changed in a good or bad way.

However, a number of difficulties, both technical and conceptual, appeared while implementing this:

- The change of statistics after text improvement could be influenced not only by better readability but also by other factors such as an added image or a fixed bug on the platform. Because of this, the decision about text changes playing a role in the improved metrics could be made incorrectly.
- Another conceptual difficulty was that the changes to texts are rarely done at the same time. There are multiple people working on it over a prolonged period of time. It means that the changes may consist of many "insignificant" improvements that together result in a substantial modification.
- For better analysis of statistical changes, it would make sense to use causal impact analysis Glymour et al. [2016], a standard procedure that the Hyperskill team employs for statistics tracking. It is a method where fluctuations of statistical metrics of a unit are compared to those of similar units, and a conclusion about the significance of a change is made based on that comparison. However, this analysis is usually made manually, there are few instruments that can perform it automatically, and what is more, it is not evident how to store and compare the results.
- The second technical difficulty, and the main one, was that I couldn't find a way to access the history of edits automatically. It was due to the fact that this information was stored not on Hyperskill but in another company's backend, to which I didn't have access.

Even though in this research, due to the mentioned difficulties, I decided not to dive into this, it may be a good topic to continue the investigation.

4.8.2 Other content on the platform

The methods explored in the study dealt with the theoretical parts of the topics. A topic, however, consists of a theory and several *tasks*: usually short texts composed of one or several sentences. The readability of short texts is a more challenging task but it can be a natural course for further research, especially since for tasks we can also employ the statistical information from the platform.

Apart from a topic, there is another educational unit on Hyperskill: a *project*. It is a complex practical task separated into several stages, completing which the student writes a program that becomes more advanced with each step. Every stage contains instructions on what the output should look like. In the current research, I did not work with projects, since their format and style differ a lot from the theoretical parts of topics, and I decided to focus on the latter. However, projects' descriptions are also textual information that should be understandable and that can be checked with the help of readability methods.

4.8.3 Better understand the effect of particular features

Similarly to the previous idea of increasing the number of analyzed units on the platform, qualitative research could also be extended. Most importantly, it could be investigated more closely what features influence the results of the machine learning systems the most. Some analysis in this direction has been done, but

I didn't analyze the effects of separate features. Another idea could be to add more various features and investigate their contributions as well.

4.8.4 Focus on the users

Finally, throughout this research, I focused on the average user statistics of texts, as well as one readability assessment per topic. However, they can be largely influenced by particular users. Thus, another possibility would be to analyze various sub-groups of users (e.g. beginners VS advanced students, or new and old platform users), or even predict readability and metrics such as the average like for a specific user, based on their history on the platform.

However, even though many directions of further research are available, the main goal of the work was achieved: I developed a process for the Hyperskill team to trace poorly-written topics among the published ones and for new topics – to predict if a topic will be well-liked and easy to complete.

Conclusion

This work regarded the task of measuring readability of technical texts from the perspective of an online educational platform on Computer Science. This fact shaped the limitations and opportunities of my research.

Some of the challenges were as follows:

- There was no gold data available on the readability of texts, as would be the case if I explored this topic on any of the available corpora for this purpose. I solved this issue by finding parameters that could reflect texts' readability.
- The size of the corpus was limited to around 1000 texts, which did not allow me to use the most recent approaches connected to Neural Networks. However, I was still able to train classical machine learning algorithms on this data.

The advantages, in turn, included the following:

- For each text, user statistics were available, some of which I could use as a proxy for readability and as training features for machine learning systems.
- There were user comments for many topics, which allowed me to track students' opinions on the texts.

These aspects specified the setting of my work. Before concluding it, I would like to sum up the main milestones of my research.

In the first part of the thesis, I tried to accommodate the standard readability formulas. I calculated nine various scores and studied their correlation with the statistical characteristics of texts. In this setting, the user statistics, and completion time in particular, were seen as reflecting readability. I enhanced the new Dale-Chall readability formula by creating custom vocabularies for every text based on the material that the student should have completed. However, all the correlations were low, and the conclusion was made that the readability of texts on the platform was a more complex parameter than such surface characteristics were able to capture.

In the second part of the research, I tried to predict several parameters with the help of machine learning algorithms. Firstly, I used linguistic and meta features, probability from a language model, and LSA topic weights to predict the average like, completion rate, and completion time for the texts. The best results were achieved when using all of the features and corresponded to roughly 10% of the parameters' ranges for the average like and the completion time, and to 16% of the range for the completion rate. Secondly, I tried several rule-based methods of dividing the topics into well- and poorly-readable ones that took comments into account. After selecting the best-performing one, which was based solely on the presence of specific words in the comments, I trained binary classification models to split the topics according to these groups. The results were unsatisfactory, leading me to a conclusion, similarly to that of the first part,

that the used features were unable to capture readability expressed as such topic division.

The final system that I present as the result of my work relies on the rule-based algorithm of comments tracking in order to identify poorly-written texts among the published ones. For new texts, the pipeline suggests predicting their average like, completion rate and time, and marking the topics as requiring attention from the Hyperskill team if their anticipated metrics do not lie within the acceptable ranges.

To conclude, the current work was aimed at solving a practical goal, which it successfully achieved. However, the findings made in the thesis pave the way for future research.

Bibliography

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

Sandra Aluisio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. Readability assessment for text simplification. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9, Los Angeles, California, June 2010. Association for Computational Linguistics. URL `https://aclanthology.org/W10-1001`.

Bonnie B Armbruster. The problem of "inconsiderate text.". *Comprehension instruction: Perspectives and suggestions*, pages 202–217, 1984.

Ion Madrazo Azpiazu and Maria Soledad Pera. Multiattentive Recurrent Neural Network Architecture for Multilingual Readability Assessment. *Transactions of the Association for Computational Linguistics*, 7:421–436, 07 2019. ISSN 2307-387X. doi: 10.1162/tacl_a_00278.

Yahia Baashar, Gamal Alkawsi, Nor'ashikin Ali, Hitham Alhussian, and Hussein T Bahbouh. Predicting student's performance using machine learning methods: A systematic literature review. In *2021 International Conference on Computer & Information Sciences (ICCOINS)*, pages 357–362. IEEE, 2021.

John C Begeny and Diana J Greene. Can readability formulas be used to successfully gauge difficulty of reading materials? *Psychology in the Schools*, 51 (2):198–215, 2014.

John R. Bormuth. Readability: A new approach. *Reading Research Quarterly*, 1 (3):79, 1966. doi: 10.2307/747021. URL `https://doi.org/10.2307/747021`.

Aissa Boudjella, Mukti Sharma, and Deepti Sharma. Non-native english speaker readability metric: Reading speed and comprehension. *Journal of Applied Mathematics and Physics*, 5(6):1257–1268, 2017.

Jeanne Sternlicht Chall and Edgar Dale. *Readability revisited: The new Dale-Chall readability formula*. Brookline Books, 1995.

Charles H Clark. Assessing comprehensibility: The PHAN system. *Read. Teach.*, 34(6):670–675, 1981.

J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960. ISSN 0013-1644. doi: 10.1177/001316446002000104. URL `http://epm.sagepub.com/cgi/content/refs/20/1/37`.

Meri Coleman and Ta Lin Liau. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283, 1975.

Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. Automatic generation of natural language explanations. In *Proceedings of the 23rd international conference on intelligent user interfaces companion*, pages 1–2, 2018.

Scott A Crossley, David F Dufty, Philip M McCarthy, and Danielle S McNamara. Toward a new readability: A mixed model approach. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 29, 2007.

Saman Daneshvar and Diana Inkpen. Gender identification in twitter using n-grams and lsa. In *Proceedings of the Ninth International Conference of the CLEF Association (CLEF 2018)*, 2018.

François Daoust, Léo Laroche, and Lise Ouellet. Sato-calibrage: présentation d'un outil d'assistance au choix et à la rédaction de textes pour l'enseignement. *Revue québécoise de linguistique*, 25(1):205–234, 1996. doi: https://doi.org/10.7202/603132ar.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

Felice Dell'Orletta, Simonetta Montemagni, and Giulia Venturi. READ–IT: Assessing readability of Italian texts with a view to text simplification. In *Proceedings of the Second Workshop on Speech and Language Processing for Assistive Technologies*, pages 73–83, Edinburgh, Scotland, UK, July 2011. Association for Computational Linguistics. URL `https://aclanthology.org/W11-2308`.

Carmine DiMascio. py-readability-metrics. `https://github.com/cdimascio/py-readability-metrics`, 2019.

W. Dubay. The principles of readability. *CA*, 92627949:631–3309, 01 2004.

E B Entin and G R Klare. Relationships of measures of interest, prior knowledge, and readability to comprehension of expository passages. *Advances in Reading/Language Research*, 3:9–38, 1985.

Eileen B Entin. Relationships of measures of interest, prior knowledge, and readability to comprehension of expository passages. 1981.

Warren Fass and Gary M Schumacher. Effects of motivation, subject activity, and readability on the retention of prose materials. *J. Educ. Psychol.*, 70(5): 803–807, 1978.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. A comparison of features for automatic readability assessment. 2010.

R Flesch. A new readability yardstick. *J. Appl. Psychol.*, 32(3):221–233, June 1948.

Thomas François. Readability: a one-hundred-year-old field still in his teens, July 2015a.

Thomas François. When readability meets computational linguistics: A new paradigm in readability. *Revue Française de Linguistique Appliquée*, 20:79–97, 11 2015b. doi: 10.3917/rfla.202.0079.

Thomas François, Adeline Müller, Eva Rolin, and Magali Norré. Amesure: A web platform to assist the clear writing of administrative texts. In *AACL*, 2020.

Perla B Gámez and Nonie K Lesaux. Early-adolescents' reading comprehension and the stability of the middle school classroom-language environment. *Developmental Psychology*, 51(4):447, 2015.

Madelyn Glymour, Judea Pearl, and Nicholas P Jewell. *Causal inference in statistics: A primer*. John Wiley & Sons, 2016.

W S Gray and B E Leary. *What makes a book readable: With special reference to adults of limited reading ability*. University of Chicago Press, Chicago, Ill, 1935.

R. Gunning. *The technique of clear writing*. Toronto, McGraw-Hill, 1952.

Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020. doi: 10.1038/s41586-020-2649-2.

Michael Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 460–467, Rochester, New York, April 2007. Association for Computational Linguistics. URL `https://aclanthology.org/N07-1058`.

Alyson Hill and LV Scharff. Readability of websites with various foreground/background color combinations, font types and word styles. In *Proceedings of 11th National Conference in Undergraduate Research*, volume 2, pages 742–746, 1997.

Leon C Hull. Measuring the readability of technical writing. In *Proceedings of the 26th International Technical Communications Conference*, pages E79–E84, 1979.

Walayat Hussain, Osama Sohaib, Atiq Ahmed, and M Qasim Khan. Web readability factors affecting users of all ages. *Australian Journal of Basic and Applied Sciences*, 2011.

Clayton J. Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In Eytan Adar, Paul Resnick, Munmun De Choudhury, Bernie Hogan, and Alice H. Oh, editors, *ICWSM*. The AAAI Press, 2014. ISBN 978-1-57735-659-2. URL `http://dblp.uni-trier.de/db/conf/icwsm/icwsm2014.html#HuttoG14`.

Milton D Jacobson. Reading difficulty of physics and chemistry textbooks. *Educ. Psychol. Meas.*, 25(2):449–457, July 1965.

Susan Kemper. Measuring the inference load of a text. *Journal of Educational Psychology*, 75:391–401, 06 1983. doi: 10.1037/0022-0663.75.3.391.

Richard P Kern. Usefulness of readability formulas for achieving army readability objectives: Research and state-of-the-art-applied to the army's problem. Technical report, ARMY RESEARCH INST FOR THE BEHAVIORAL AND SOCIAL SCIENCES ALEXANDRIA VA, 1980.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch, 1975.

Walter Kintsch. On modeling comprehension. *Educational Psychologist*, 14:3–14, 1979.

Walter Kintsch and Douglas Vipond. Reading comprehension and readability in educational practice and psychological theory. 1979.

Walter Kintsch and Douglas Vipond. Reading comprehension and readability in educational practice and psychological theory. *Perspectives on learning and memory*, pages 329–365, 2014.

G. R. Klare. *The measurement of readability*. Ames: Iowa State University Press, 1963.

G R Klare and K L Smart. Analysis of the readability level of selected usafi instructional materials. *The journal of educational research*, 1973.

George R Klare. A second look at the validity of readability formulas. *Journal of reading behavior*, 8(2):129–152, 1976.

George R Klare, James E Mabry, and Levarl M Gustafson. The relationship of style difficulty to immediate retention and to acceptability of technical material. *J. Educ. Psychol.*, 46(5):287–295, May 1955.

Miroslav Kubát, Vladimír Matlach, and Radek Čech. Quita. *Quantitative Index Text Analyzer. Lüdenscheid: RAM-Verlag*, 2014.

Timo Lenzner. Are readability formulas valid tools for assessing survey question difficulty? *Sociological Methods & Research*, 43(4):677–698, 2014.

Bertha A Lively and Sidney L Pressey. A method for measuring the vocabulary burden of textbooks. *Educational administration and supervision*, 9(7):389–398, 1923.

Steven Loria. textblob documentation. *Release 0.15*, 2, 2018.

Kelly Marchisio, Jialiang Guo, Cheng-I Lai, and Philipp Koehn. Controlling the reading level of machine translation output. In *Proceedings of Machine Translation Summit XVII: Research Track*, pages 193–203, 2019.

Howard Y Mcclusky. A quantitative analysis of the difficulty of reading materials. *J. Educ. Res.*, 28(4):276–282, December 1934.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

GH McLaughlin. Smog grading–a new readability formula in the journal of reading, 1969.

Danielle S. McNamara, Arthur C. Graesser, Philip M. McCarthy, and Zhiqiang Cai. *Automated Evaluation of Text and Discourse with Coh-Metrix*. Cambridge University Press, 2014. doi: 10.1017/CBO9780511894664.

Bonnie J. F. Meyer. Reading research and the composition teacher: The importance of plans. *College Composition and Communication*, 33:37–49, 1982.

Ines Montani, Matthew Honnibal, Matthew Honnibal, Sofie Van Landeghem, Adriane Boyd, Henning Peters, Paul O'Leary McCann, Maxim Samsonov, Jim Geovedi, Jim O'Regan, Duygu Altinok, György Orosz, Søren Lind Kristiansen, , Roman, Explosion Bot, Lj Miranda, Leander Fiedler, Daniël De Kok, Grégory Howard, , Edward, Wannaphong Phatthiyaphaibun, Yohei Tamura, Sam Bozek, , Murat, Mark Amery, Ryn Daniels, Björn Böing, Pradeep Kumar Tippa, and Peter Baumgartner. explosion/spacy: v3.1.6: Workaround for click/typer issues, 2022. URL `https://zenodo.org/record/1212303`.

John O'hayre. *Gobbledygook has gotta go*. US Department of the Interior, Bureau of Land Management, 1966.

Ralph H Ojemann. The reading ability of parents and factors associated with the difficulty of parent education materials. *Researchers in Parent Education. II*, 8, 1934.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Simone Primarosa. lm-scorer. `https://github.com/simonepri/lm-scorer`, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Sarah Schwarm and Mari Ostendorf. Reading level assessment using support vector machines and statistical language models. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 523–530, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219905. URL `https://aclanthology.org/P05-1065`.

Judith A Shaw. Reading problems in mathematical texts. 01 1967.

Kathleen M. Sheehan, Michael Flor, and Diane Napolitano. A two-stage approach for generating unbiased estimates of text complexity. In *Proceedings of the Workshop on Natural Language Processing for Improving Textual Accessibility*, pages 49–58, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL `https://aclanthology.org/W13-1506`.

Wade Shen, Jennifer Williams, Tamas Marius, and Elizabeth Salesky. A language-independent approach to automatic text difficulty assessment for second-language learners. Technical report, Massachusetts Inst of tech Lexington Lincoln lab, 2013.

Luo Si and Jamie Callan. A statistical model for scientific readability. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, page 574–576, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581134363. doi: 10.1145/502585.502695. URL `https://doi.org/10.1145/502585.502695`.

E. Smith. Devereaux readability index. *The Journal of Educational Research*, 54: 289–303, 1961.

Edgar A Smith and RJ Senter. *Automated readability index*, volume 66. Aerospace Medical Research Laboratories, 1967.

George Spache. A new readability formula for primary-grade reading materials. *The Elementary School Journal*, 53(7):410–413, 1953.

Alfred Stenner. Measuring reading comprehension with the lexile framework. *Fourth North American Conference on Adolescent/Adult Literacy. Washington, D.C*, 01 1996.

Kumiko Tanaka-Ishii, Satoshi Tezuka, and Hiroshi Terada. Sorting texts by readability. *Computational Linguistics*, 36:203–227, 06 2010. doi: 10.1162/coli.2010.09-036-R2-08-050.

Wilson L. Taylor. "cloze procedure": A new tool for measuring readability. *Journalism & Mass Communication Quarterly*, 30:415 – 433, 1953.

E Thorndike. Word knowledge in the elementary school. *The Teachers College Record*, 22:334–370, 1921.

Edward Lee Thorndike. *An Improved Scale for Measuring Ability in Reading...*, volume 16. 1916.

PLAIN UK. Plain english campaign. `http://www.plainenglish.co.uk/about-us.html`. Last accessed: 18-07-22.

PLAIN US. Plain language action and information network. `https://www.plainlanguage.gov/about/`. Last accessed: 18-07-22.

Sowmya Vajjala and Detmar Meurers. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 163–173, Montréal, Canada, June 2012. Association for Computational Linguistics. URL `https://aclanthology.org/W12-2019`.

Mabel Vogel and Carleton Washburne. An objective method of determining grade placement of children's reading material. *Elem. Sch. J.*, 28(5):373–381, January 1928.

Allan R Williams, Arthur I Siegel, James R Burkett, and Steven D Groff. Development and validation of an equation for predicting the comprehensibility of textual material. Technical report, APPLIED PSYCHOLOGICAL SERVICES INC WAYNE PA, 1977.

Beverly L Zakaluk and S Jay Samuels. *Readability: Its Past, Present, and Future.* ERIC, 1988.

Wei Zhang, Sui-xi Kong, Yan-chun Zhu, and Xiao-le Wang. Sentiment classification and computing for online reviews by a hybrid svm and lsa based approach. *Cluster Computing*, 22(5):12619–12632, 2019.

Xinran Zhang, Maosong Sun, Jiafeng Liu, and Xiaobing Li. Improving diversity of neural text generation via inverse probability weighting. *arXiv preprint arXiv:2103.07649*, 2021.

A. Attachments

A.1 The lists of words used in the rule-based algorithms

This attachments contains lists of words that were used for the rule-based algorithms described in section 4.3.

Adjectives: confounding, nebulous, lousy, hard, obscure, bad, scandalous, cumbersome, hideous, horrendous, difficult, imprecise, knotty, upsetting, odious, offbeat, burdensome, abominable, astonishing, dreadful, incomprehensible, schocking, inconclusive, poor, unsure, cryptic, uncertain, sad, strange, bizzare, ambiguous, questionable, perplexing, fuzzy, unacceptable, wrong, terrifying, baffling, equivocal, amateurish, horrid, tough, misleading, problematic, inaccurate, heinous, bewildering, intricate, complicated, troublesome, disastrous, appalling, opaque, dubioud, confused, arduous, terrible, horrible, horrific, complex, inexplicit, awful, barbaric, unpleasant, sophisticated, weird, puzzling, dubious, disgusting, unfortunate, confusing, heavy, unwieldy, hazy, lax, vague, convoluted, atrocious, unsettled, disconcerting, awkward, unclear.

Positive adjectives: coherent, comprehensible, transparent, clear, easy, readable, simple, precise, great, good, uncomplecated, intelligible, well, unambiguous, understandable, straightforward, evident, smooth, effortless, acceptable, obvious, explicit.

Verbs: trouble, mislead, realize, upset, baffle, improve, obscure, complicate, specify, develop, convolute, word, frustrate, formulate, perplex, conceive, learn, explain, enhance, grasp, puzzle, rewrite, describe, misinform, illustrate, write, depict, understand, amplify, define, elaborate, disconcert, infer, detail, represent, confound, clarify, interpret, misguide, confuse, apprehend, refine.

Nouns: instruction, discontent, effort, task, exercise, grasp, statement, definition, struggle, assignment, complication, concern, topic, dissatisfaction, hardship, understanding, difficulty, lesson, wording, project, text, readability, description, problem, phrasing, inconvenience, question, word, obstacle, comprehension, english, explanation, frustration, apprehension, issue, direction, trouble, language.

Words about writing, reading, and understanding: clarify, task, grasp, english, explain, specify, detail, language, realize, project, assignment, comprehension, statement, describe, amplify, enhance, interpret, phrasing, depict, elaborate, improve, explanation, conceive, refine, represent, wording, topic, rewrite, text, lesson, understanding, definition, define, apprehension, write, infer, formulate, direction, exercise, instruction, learn, understand, readability, word, develop, illustrate, apprehend, description.

Words about unclearness: convolute, confound, perplex, struggle, hardship, baffle, upset, trouble, disconcert, problem, issue, concern, confuse, complicate, mislead, effort, complication, misguide, question, frustration, obstacle, difficulty, discontent, dissatisfaction, inconvenience, frustrate, obscure, puzzle, misinform, confounding, nebulous, lousy, hard, obscure, bad, scandalous, cumbersome, hideous, horrendous, difficult, imprecise, knotty, upsetting, odious, offbeat, burdensome, abominable, astonishing, dreadful, incomprehensible, schocking, inconclusive, poor, unsure, cryptic, uncertain, sad, strange, bizzare, ambiguous,

questionable, perplexing, fuzzy, unacceptable, wrong, terrifying, baffling, equivocal, amateurish, horrid, tough, misleading, problematic, inaccurate, heinous, bewildering, intricate, complicated, troublesome, disastrous, appalling, opaque, dubioud, confused, arduous, terrible, horrible, horrific, complex, inexplicit, awful, barbaric, unpleasant, sophisticated, weird, puzzling, dubious, disgusting, unfortunate, confusing, heavy, unwieldy, hazy, lax, vague, convoluted, atrocious, unsettled, disconcerting, awkward, unclear.

A.2 Distributions of comments classified as positive or negative

As described in the 4.3, different versions of the rule-based algorithm used different approaches to classification of comments. This section illustrates how the comments were classified. The value -1 for the comments means that they are considered negative, 0 means that they could not be classified as either positive or negative, 1 corresponds to positive ones, and 0.5 implies that there were no "trigger words" found.

Figure A.1: Distribution for the first version

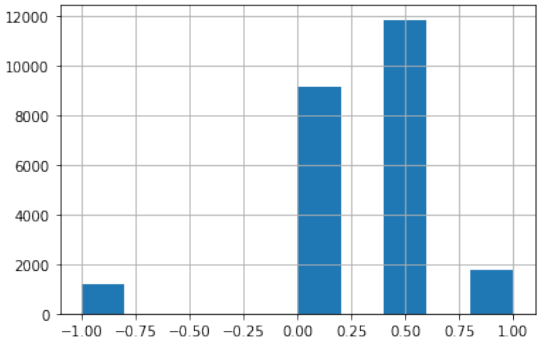


Figure A.2: Distribution for the second version

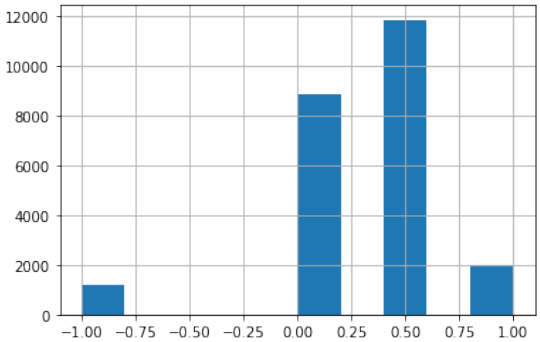


Figure A.3: Distribution for the third version

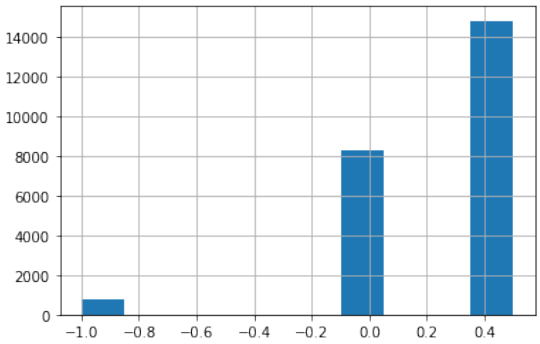


Figure A.4: Distribution for the fourth version

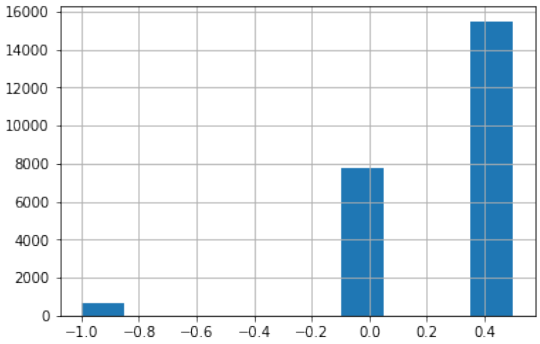


Figure A.5: Distribution for the fifth version

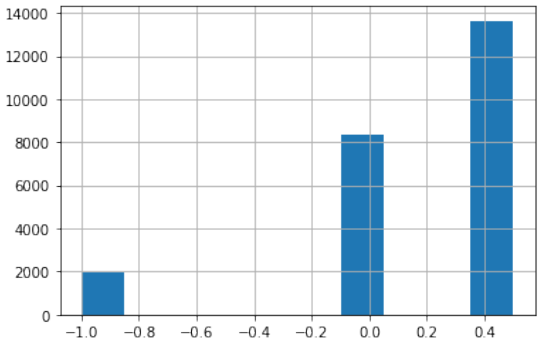
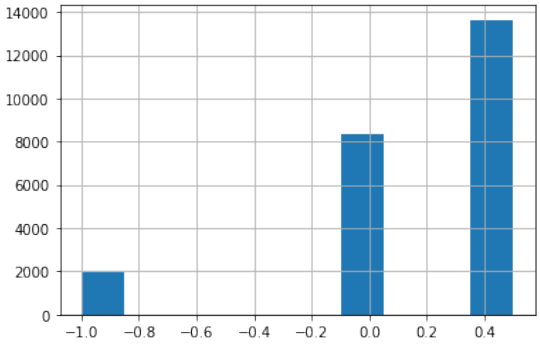


Figure A.6: Distribution for the fifth version with the likes



A.3 Distributions of various statistical parameters across all topics

This attachment contains graphs with distributions for all available statistical parameters described in detail in section 2.4 across topics with at least 20 completions.

Figure A.7: Distribution for seconds to complete

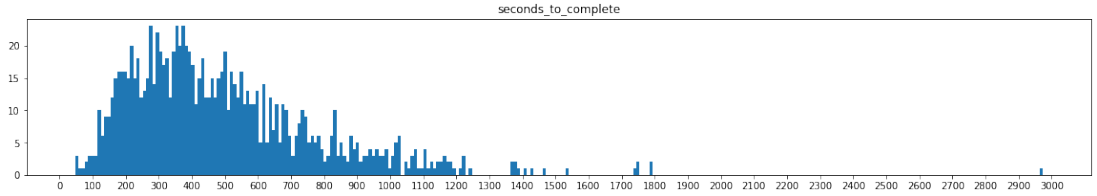


Figure A.8: Distribution for the average like

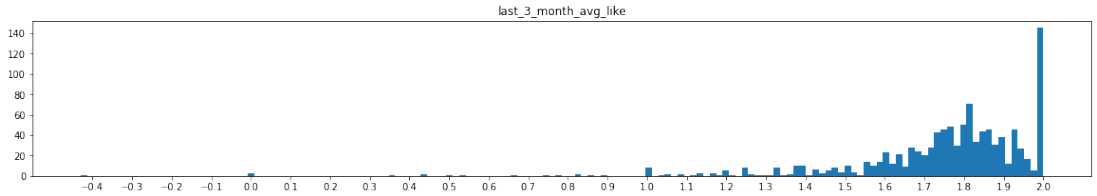


Figure A.9: Distribution for the like count

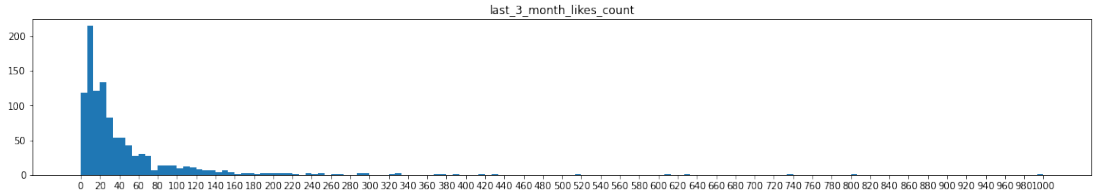


Figure A.10: Distribution for the completion rate

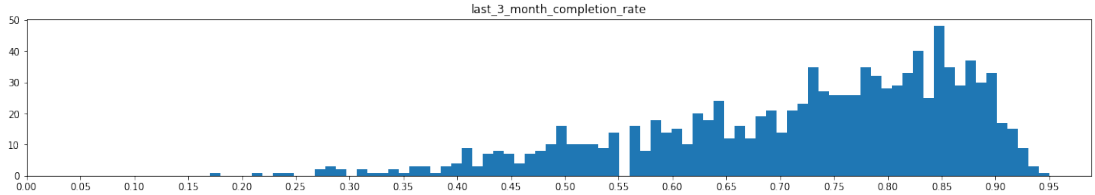


Figure A.11: Distribution for the number of users who completed the step

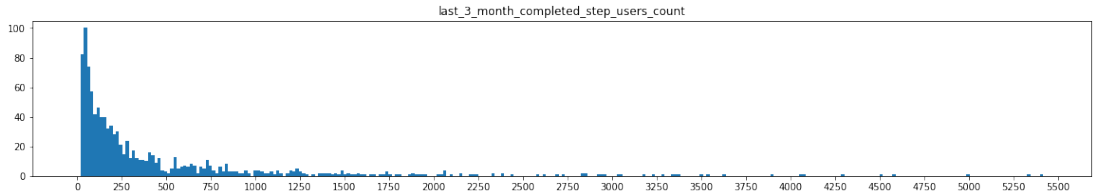


Figure A.12: Distribution for the *topic* completion rate

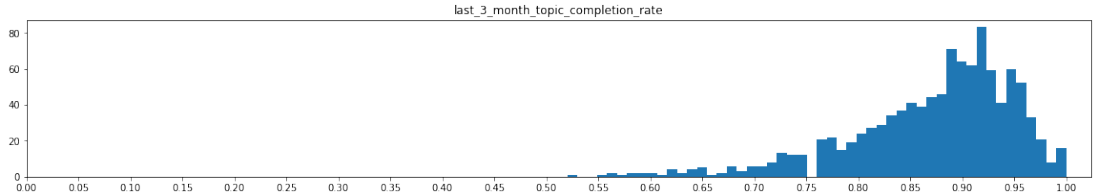


Figure A.13: Distribution for the topic completion count

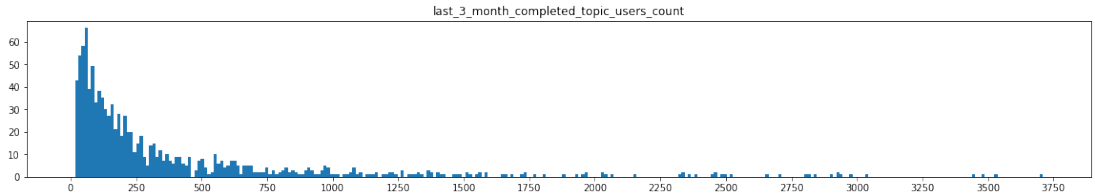


Figure A.14: Distribution for the percent of users who got back to theory

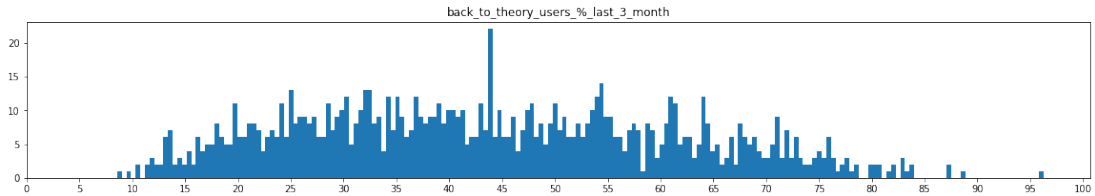


Figure A.15: Distribution for the average number of getting back to theory for one user session