TakeLab

Laboratorij za analizu teksta i inženjerstvo znanja
Text Analysis and Knowledge Engineering Lab
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER THESIS n. 2036

Predicting Text Readability using Natural Language Processing Methods

Robert Injac

Zagreb, July 2019

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING
MASTER THESIS COMMITTEE

Zagreb, 8 March 2019

MASTER THESIS ASSIGNMENT No. 2036

Student: **Robert Injac (0036486492)**
Study: Computing
Profile: Computer Science

Title: **Predicting Text Readability using Natural Language Processing Methods**

Description:

Text readability is defined as the ease with which a reader can understand a written text. Predicting readability of a text has many uses, especially in education, where it is important to provide students with texts they can easily understand. A number of formulas have been developed that calculate readability using text features, such as average sentence length or the number of uncommon words as inputs. In recent years, there have been attempts to calculate text readability using novel natural language processing methods, typically ones based on machine learning.

The topic of the thesis are models for predicting text readability using natural language processing. Do a literature survey on existing methods for calculating readability based on formulas or machine learning. Develop a model for predicting readability of texts in English language using natural language processing based on machine learning. Carry out an experimental evaluation of the model, a comparison against the existing models, and a statistical analysis of the results. All references must be cited, and all source code, documentation, executables, and datasets must be provided with the thesis.

Issue date: 15 March 2019
Submission date: 28 June 2019

Mentor:

Associate Professor Jan Šnajder, PhD

Committee Chair:

Assistant Professor Marko Čupić, PhD

Committee Secretary:

Associate Professor Tomislav Hrkać, PhD

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 8. ožujka 2019.

DIPLOMSKI ZADATAK br. 2036

Pristupnik: **Robert Injac (0036486492)**
Studij: Računarstvo
Profil: Računarska znanost

Zadatak: **Procjena čitljivosti teksta postupcima obrade prirodnog jezika**

Opis zadatka:

Čitljivost teksta definirana je kao lakoća kojom čitatelj može razumjeti čitani tekst. Procjena čitljivosti teksta ima niz primjena, posebno u obrazovanju, gdje je važno učenicima predočiti tekstove koje mogu lako razumjeti. Razvijeno je mnogo formula koje izračunavaju čitljivost na temelju značajki teksta, poput prosječne duljine rečenice ili broja rijetkih riječi. Posljednjih godina bilo je pokušaja da se čitljivost izračuna koristeći novije metode obrade prirodnog jezika, koje su obično temeljene na strojnom učenju.
Tema rada jesu modeli za procjenu čitljivosti teksta temeljeni na obradi prirodnog jezika. Potrebno je napraviti pregled postojećih metoda izračuna čitljivosti koje su temeljene na formulama ili strojnom učenju. Razviti model za procjenu čitljivosti tekstova na engleskome jeziku koji će koristiti obradu prirodnog jezika temeljnu na strojnome učenju. Provesti eksperimentalno vrednovanje modela, usporedbu s postojećim modelima i statističku analizu rezultata. Radu priložiti izvorni i izvršni kod razvijenog sustava, skupove podataka i programsku dokumentaciju te citirati korištenu literaturu.

Zadatak uručen pristupniku: 15. ožujka 2019.
Rok za predaju rada: 28. lipnja 2019.

Mentor:

Izv. prof. dr. sc. Jan Šnajder

Djelovođa:

Izv. prof. dr. sc. Tomislav Hrkać

Predsjednik odbora za diplomski rad profila:

Doc. dr. sc. Marko Čupić

TABLE OF CONTENTS

1. Introduction

Written text can have many attributes, but one of them is perhaps the most important of them all – readability. If a text is barely readable, it can be worthless: children will not learn well from convoluted textbooks, people will not read incomprehensible books, and advertisement written in complicated language will fail to attract new customers. Easy reading, on the other hand, helps learning and enjoyment. For these reasons, assessing the level of readability is of great importance.

Unfortunately, readability is not so easy to predict. The level of readability is not an objective measure, and it depends on the skill of the reader and the topic of the text. However, early researchers have noted certain features in highly readable text such as shorter sentences and usage of less complex words. Using these features, they constructed an array of formulas used for predicting readability. These formulas are easy to calculate and have been proved to be quite successful in the fields of government, teaching, publishing, the military, medicine, and business.

Formulas, however, have some downsides. They use a limited number of features and require a high amount of manual calibration for each added feature. As a result, people have been turning over to computers to help them calculate readability. Researchers in natural language processing (NLP), a field concerned with programming computers to process and analyze large amounts of natural language, have developed many methods which improve readability prediction. These methods, coupled with machine learning, can be used to calculate new features from text and construct models which predict readability from those features.

The question of which methods and which models can be used to predict readability is the topic of this thesis. Firstly, readability and the task of predicting readability will be explained in detail. Secondly, NLP methods which are used to generate features will be shown. Thirdly, traditional formulas used in readability prediction will be summarized. Fourthly, machine learning models which predict readability will be described. Last but not least, on all formulas and models a statistic analysis will be performed to find out which one is the most efficient.

2. Readability

Readability is defined as **the ease with which a reader can understand a written text.** Broadly speaking, readability depends on text's content (the syntax and semantics) and its graphical representation (such as font size, font style, line height, and line length). It this work, only the readability depending on content will be evaluated.

The uses of readability are varied. One of the first and most important uses of readability is in education. Teachers want to match the difficulty level of the material to their student's reading ability. If a student is given the textbook readable on their level, oral reading errors are decreased and reading comprehension is increased. Readability prediction methods, such as readability formulas, are used in schools and classrooms all over the world, by teachers interested in selecting stories and books for classroom use and for individual student's particular needs (Fry, 2006).

Outside of the classroom, the manufacturing business also have a need for readability. Booklets and labels are the primary method of communication between the manufacturer and the consumer, and they need to be easily readable. Other businesses such as insurance, banking, and retail need readable policies, contracts, and catalogs. In many business cases, a considerable amount of money was saved by using readability measuring to write more readable text (Fry, 1987).

Using less readable text can even get companies or organizations sued. Readability is directly involved in many court cases, and readability measures have been established as a part of contested legal proceedings. For example, in a class action suit involving national health care appeals, much of the contention involved the readability of a form letter sent out by the New York Medicare office (Fry, 1987).

Even more readability uses are found in government and publishing industry domains. Readability research has helped millions of students, newspaper readers, insurance policy buyers, readers of legal contracts, buyers of new products, and many others.

2.1. Readability Prediction

The task of assessing the readability of a text or readability prediction is a **well-studied problem in linguistics.** A system for readability prediction takes the text as an input, and gives a real number or an integer as an output. A real number represents a readability measure (usually, a larger number means a text is less readable), while the integer represents a readability level (in educational cases, this discrete level represents grade of the student; for example, level 6 means a 6th grader is able to read the text).

Usually, the text on which the prediction is made comes from a book. However, readability designations have been applied to almost every kind of prose including laws, newspaper articles, test passages, military manuals, and advertising (Fry, 2006).

History of readability prediction starts in the late 19th century. The fact that oral speech is more understandable than written text was noticed by English professor L. A. Sherman. He suggested using shorter sentences and concrete terms to help people make sense of what is written (Sherman, 1893). Russian writer Nikolai A. Rubakin published a study of over 10,000 texts written by everyday people, and collected 1,500 words from that collection which he thought most people understood. He found that the main blocks to comprehension are unfamiliar words, and that people are in the need of books written at a level they could grasp (Choldin, 1979).

The 20th century brought the birth of a powerful method for readability prediction – **the readability formula.** Readability formulas work in the following way: firstly, some numeric features are extracted from the text (one of those features could be the average number of words per sentence), and then those features are put in as variables into a fixed formula which outputs a readability measure. Readability formulas are quite simple to work with and give effective results.

The first readability formula was published in 1923 by Lively and Pressey (Lively and Pressey, 1923). However, the most used formulas were made in 1940s. These are the Dale-Chall formula, the Flesch reading ease, and the Gunning fog index. The readability formulas will be described and explained in detail in Chapter 4.

The field of readability has changed a lot in the 21st century. Firstly, **natural language processing methods** are more and more used to extract complex features from the text. Secondly, **machine learning models** are trained on these features to predict readability. The NLP methods and the machine learning models will be explained in detail in their respective chapters.

2.2. Readability Evaluation Datasets

One of the most important things in constructing a formula or an algorithm for readability prediction is **the dataset on which it is evaluated**. The dataset should be of adequate size (the larger the better) and should be topical (readability of newspaper articles should not be evaluated on a dataset of math textbooks).

Traditionally, readability was evaluated using **graded passages**. The graded passage is a basic unit of evaluation in which a paragraph is assigned a grade level, typically by experts at an educational organization or government entity (Collins-Thompson, 2014). These graded passages are then used by researchers to form a corpus for evaluation of readability prediction. Unfortunately, it is often the case that existing readability measures are used to calibrate graded passages, and so when evaluating new readability measures, there may be a bias in favor of those or similar measures that were used to calibrate the passages.

Recently, most datasets used in readability prediction **are derived from textbooks, newspaper articles and web pages** written for different target audiences (Vajjala and Lučić, 2018). Different audiences can be children of different age or foreign language learners of a different skill level. The readability level is then assumed. For example, the articles written for younger children are assumed to be more readable than ones written for older children. Textbooks written for A1 learners of English are assumed to have more readable passages then textbooks written for B2 learners.

In general, many studies have created their own datasets. Access to these research datasets can be granted by contacting the authors. For copyright and licensing reasons some datasets are restricted from being made freely available. At the time of writing there is still a lack of significantly-sized, freely available, high-quality dataset for computational readability evaluation (Collins-Thompson, 2014).

2.3. WeeBit Dataset

WeeBit is a readability evaluation dataset created by Vajjala and Meurers in their 2012 paper (Vajjala and Meurers, 2012). They used a **combined corpus of newspaper articles** from *WeeklyReader* and *BBC-Bitesize* to develop a **dataset with five grade levels, based on age groups.** Each grade level can be used as one readability level, with levels belonging to younger children having higher, and levels belonging to older children having lower readability. The dataset has been used in numerous later readability studies, and continues to be one of the largest datasets for readablity evaluation.

WeeklyReader was a weekly educational classroom magazine designed for children. It was merged with its new owner, *Scholastic News*, in 2012. The articles were targeted at four grade levels (Level 2, Level 3, Level 4, and Senior), corresponding to children between ages 7–8, 8–9, 9–10, and 9–12 years. They covered a wide range of non-fiction topics. The authors obtained permission to use the graded magazine articles and downloaded the archives in November 2011. *WeeklyReader* issued additions to the actual articles which included teacher guides, student quizzes, images and brain teaser games, which the authors did not include in the corpus.

BBC-Bitesize is a free online study support resource for school-age students in the United Kingdom. It has articles classified into four grade levels (KS1, KS2, KS3 and GCSE), corresponding to children between ages 5–7, 8–11, 11–14 and 14–16 years. The Bitesize corpus is freely available on the web, and the authors crawled it in 2009. Considering most of the articles at KS1 level consisted of images and flash files with little text, it was excluded.

To make the WeeBit dataset **the authors used Level 2, 3 and 4 from *WeeklyReader* and KS3 and GCSE from *BBC-Bitesize*.** This combined corpus covers learners aged 7 to 16 years. The level Senior from WeeklyReader and level KS2 from BBC-Bitesize overlaped, and were excluded from the combined corpus.

The WeeBit dataset is used in this work as the evaluation corpus. The permission was granted from the authors to use it for research purposes. The 5 grade levels are converted into 5 readability levels, level 0 to level 4. The exact mapping from grade levels to readability levels can be observed in Table 2.1. From now on, this work will always refer to readability levels 0 to 4, and not the grade levels. Articles from level 0, which are made for children aged 7-8, are assumed to have the highest readability, while each higher level is assumed to have lower and lower readability, with level 4 having it lowest.

Readability level	Grade level	Age (in years)	N. of articles (original)	N. of articles (processed)
level 0	Level 2	7-8	629	610
level 1	Level 3	8-9	801	788
level 2	Level 4	9-10	814	798
level 3	KS3	11-14	644	643
level 4	GCSE	14-16	3500	800

Table 2.1: WeeBit dataset

The original dataset contains 6388 articles, half of them belonging to level 4. While structured, the dataset is not clean and contains noise such as missing texts, non-English texts and non-content lines. Before using, the dataset was processed and cleaned. **The following actions were performed:**

1. The original dataset was divided into 5 folders representing grade levels, each folder having articles as separate textual files. This was converted into a Pandas dataframe (McKinney, 2010) which has two columns: Text (article text) and Level (readability level). Each row represents one article.

2. To clean the dataset, several actions were performed. All newlines in text were replaced by spaces. The following was removed: empty texts, duplicate texts, non-English texts, non-content lines (licences and JavaScript warnings).

3. In the original dataset, there is around 5 times more articles for level 4 than for other levels. To prevent class imbalance, only a number of examples from level 4 class are used. In other words, undersampling was performed on level 4, and 800 articles were sampled from it.

4. For evaluation of machine learning models, which will be explained in detail later, the dataset is divided into a train and test set. 80% of the examples form the train set, while the other 20% form the test set. The distribution of levels for each set is kept approximately equal.

5. Finally, the whole dataset, train and test sets are each saved into a CSV file. From this format they can be easily read back into a Pandas dataframe format for usage in Python scripts.

The processed dataset is used for all evaluations in this work. The number of articles per each level before and after processing can be seen in Table 2.1. The code used in processing and cleaning the dataset which utilizes the described steps is available online.[1]

[1] `https://github.com/RobertInjac/Master-thesis/blob/master/data/dataset_preparation.py`

3. Natural Language Processing

Natural Language Processing (NLP) is usually defined as a field concerned with **computational techniques for automatic analysis and representation of human language** (Cambria and White, 2014). It is a subfield of computer science, information engineering, and artificial intelligence.

The **history of NLP** begins in the 1950s. The following overview is made according to (Nadkarni et al., 2011). NLP started as the intersection of artificial intelligence and linguistics. It relied purely on symbolic, hand-crafted rules. These simplistic approaches were defeated by *homographs* (identically spelled words with multiple meanings) and metaphor. In an (apocryphal) famous example of English-to-Russian translation, the Biblical sentence *the spirit is willing, but the flesh is weak* was translated to *the vodka is agreeable, but the meat is spoiled.*

These problems led to a fundamental reorientation, which started in the 1980s. The reorientation can be summarized as follows:

- Simple approximations were replaced by deep analysis.
- Evaluation became more rigorous.
- Machine-learning methods that used probabilities became prominent.
- Large, annotated bodies of text (corpora) were employed to train machine-learning algorithms

This reorientation resulted in the birth of statistical NLP. **Statistical NLP** is formally defined as a part of NLP which comprises all quantitative approaches to automated language processing, including probabilistic modeling, information theory, and linear algebra (Manning et al., 1999).

Statistical NLP approaches give good results in practice due to a simple reason: by learning with real-world data, they utilize the most common cases. The more abundant and representative the data, the better they get.

3.1. Common NLP Tasks

There are many commonly researched tasks in NLP. Some of those tasks have direct real-world applications, while others are mostly referred as subtasks that are used in the process of solving larger tasks. Most NLP tasks are closely intertwined, so there is no unique division of them.

The following listing of NLP tasks is based on (Nadkarni et al., 2011) and (Kao and Poteet, 2007). The tasks are divided into two categories: low-level and high-level.

Most common **low-level NLP tasks are:**

1. **Sentence boundary detection**

   The task of deciding where sentences begin and end. Abbreviations, decimal points, email adresses, and similar constructs complicate this task.

2. **Tokenization**

   The task of dividing a sentence into its component tokens (words and punctiation). In text, tokens often contain characters typically used as token boundaries: for example, hyphens and forward slashes.

3. **Part-of-speech (POS) tagging**

   The task of classifying a word into a particular part of speech class (noun, verb, and so on) based on both its definition and its context. In English, there are problems with homographs ('set') and gerunds (verbs ending in 'ing' that are used as nouns).

4. **Stemming and lemmatization**

   These are two related tasks. Stemming is the task of reducing inflected or derived words to their word stem. The word stem does not need to be a proper root; it is enough that related words map to the same stem. Lemmatization, on the other hand, is the task of mapping inflected or derived words to their cannonical form (lemma). Considering that this process depends on the context of the word, lemmatization is much more complex task than stemming.

5. **Shallow and deep parsing**

   The task of parsing is identifying phrases and other constituents from tokens. Shallow parsing chunks each token to a specific constituent, while deep parsing builds a complete constituency-based tree (parse tree) for a sentence. An example of the difference between shallow and deep parsing is shown in Figure 3.1.

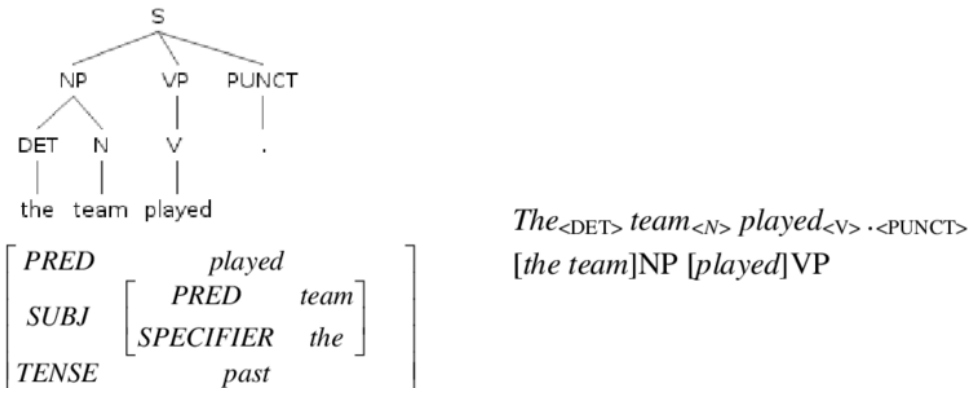

Figure 3.1: Difference between deep (left) and shallow (right) parsing of the sentence *The team played.* The shallow parser has performed POS tagging and chunking of noun and verb phrases (NPs and VPs), while the deep parser has produced a complete parse tree and has, in addition, offered information about the grammatical roles of individual words (whether they are predicates, subjects, and so on). Taken from (Kakkonen, 2007)

Higher-level NLP tasks build on low-level tasks and are usually problem-specific. Commonly included are:

1. **Named entity recognition (NER)**

   The task of identifying specific word or words ('entities') and categorizing them – for example, as persons, locations or organizations. This task often uses parsing in the search for entities (ex. nouns phrases are more likely to be entites).

2. **Relationship extraction**

   The task of determining relationships between entities or events, such as *causes* and *occurs with*. NER is used to find entities before performing this task.

3. **Sentiment analysis**

   The task of extraction and quantification of affective states and subjective information. Parsing can be used for negation handling (*I do not dislike*).

4. **Machine translation**

   The task of translating a text from one language to another. In the most simple word-for-word translation, lemmatization is needed. For more complex machine translation tasks, parse trees can be used to match constituents.

3.2. Feature Extraction Using NLP

Feature extraction is a **process which starts with an initial set of data and derives features from it.** The features are intended to be informative and non-redundant for the task at hand. The resulting features can be used for better human interpretation or as an input to a system. In machine learning, the aim of the process is facilitating the subsequent learning and generalization steps.

In this work, **statistical NLP methods will be utilized for feature extraction.** The inital set of data in this case are the article texts of the WeeBit dataset. The goal of the feature extraction step is to extract features related to readability from these texts. Systems for readability prediction (both formulas and machine learning models) which are constructed in this work do not take text as an input, but these extracted features. This will make those systems much more simple and (hopefully) efficient than they would be if they were to take the entire text as an input.

The first challenge in feature extraction is **choosing which features to extract.** The most informative features are those on which readability depends the most. Thankfully, a lot of research was done in this topic and researchers constructed hundreds of features (Collins-Thompson, 2014). Of course, many of those features are very similar, some being almost identical (ex. percentage of words longer than 8 characters and percentage of words longer than 10 characters).

The features used in this work are modeled on features used in (François and Miltsakaki, 2012). That work uses 46 features divided into 26 classical and 20 non-classical features. **Classical features** are common features used for readability prediction which require simple NLP methods to extract them. Most of them have been used from the early days of readability prediction. **Non-classical features**, on the other hand, require more complex NLP techniques like parse trees for their extraction. They have only started to be used recently with advancements in NLP research.

This work also uses the division of classical and non-classical features. **The decision was made to use 12 classical and 9 non-classical features, totalling 21 features.** The full list of features with their descriptions is given in Table 3.1. Most of the features are not direct copies of the mentioned work, but are similar in idea. Some are modeled after (Balakrishna, 2015), and some are invention of the author. The decision to use a relatively small amount of features (21 compared to 46 of the mentioned work) was made because a large number of features in the mentioned work were either too similar to another feature or not usable for the English language (the mentioned work used French).

Family	Tag	Description
Classical	Avg_words_per_sentence	Average number of words per sentence
	Avg_syllables_per_word	Average number of syllables per word
	Complex_word_percent	Percentage of words with 3 or more syllables
	Difficult_word_percent	Percentage of words not in Dale-Chall list
	Long_sent_percent	Percentage of sentences longer than 25 words
	Long_word_percent	Percentage of words longer than 8 characters
	Avg_letters_per_word	Average number of letters per word
	Comma_percent	Percentage of sentences with a comma
	Proper_noun_percent	Percentage of proper nouns
	Noun_percent	Percentage of nouns + proper nouns
	Pronoun_percent	Percentage of pronouns
	Conj_percent	Percentage of conjunctions
Non-classical	NP_per_sent	NPs (noun phrase) / num of sentences
	VP_per_sent	VPs (verb phrase) / num of sentences
	PP_per_sent	PPs (prepositional phrase) / num of sentences
	SBAR_per_sent	SBARs (subordinate clause) / num of sentences
	SBARQ_per_sent	SBARQs (question) / num of sentences
	avg_NP_size	Average lenght of an NP
	avg_VP_size	Average lenght of an VP
	avg_PP_size	Average lenght of an PP
	avg_parse_tree	Average height of a parse tree

Table 3.1: List of features

3.2.1. Extraction of Classical Features

It has been mentioned earlier that most classical features have been used for readability prediction since the early days. Long before people used computers, they extracted these features manually and used them as inputs of readability formulas. Due to their simplicity, their extraction using the computer does not require advanced NLP methods.

The NLP methods needed to extract classical features are the following:

1. Sentence boundary detection
2. Tokenization
3. POS tagging

4. Syllabification (Hyphenation)

The tasks of sentence boundary detection, tokenization and POS tagging have been explained earlier (in Section 3.1). Sentence boundary detection and tokenization is used to extract features such as *average number of words per sentence* and *percentage of sentences with a comma*. POS tagging is used for POS-related features such as *percentage of proper nouns* and *percentage of conjunctions*.

The task of **syllabification** (also called hyphenation for written language) is the separation of a word into syllables. This task is difficult since syllabification in English does not systematically follow general rules (Trogkanis and Elkan, 2010). Syllabification is needed for the features *average number of syllables per word* and *percentage of words with 3 or more syllables.*

Sentence boundary detection, tokenization and POS tagging have been implemented using *spaCy*.[1] ***spaCy*** is an open-source software library for NLP. It offers powerful statistical models which were used to perform the before-mentioned tasks. Syllabification has been implemented using *PyHyphen*.[2] ***PyHyphen*** is a Python interface to the syllabification C library used in software such as *LibreOffice* and the *Mozilla* suite.

Complete code used in the extraction of classical features is available online.[3]

3.2.2. Extraction of Non-Classical Features

In contrast to classical features, extraction of non-classical features is more complex and therefore requires the use of more advanced NLP methods. Specifically, **the extraction of non-classical features in this work requires deep parsing.** Deep parsing produces a parse tree for every sentence of the text, and from these parse trees features such as *average number of PPs (prepositional clauses) per sentence* and *average height of a parse tree* are extracted.

Deep parsing was implemented using ***benepar***.[4] It is a Python implementation of a deep parsing algorithm described in (Kitaev and Klein, 2018). *benepar* can be used as a plugin to *spaCy*, which is how it was used in this work. Complete code used in the extraction of non-classical features is available online.[5]

[1] `https://spacy.io/`

[2] `https://pypi.org/project/PyHyphen/`

[3] `https://github.com/RobertInjac/Master-thesis/blob/master/features/classic_features.py`

[4] `https://pypi.org/project/benepar/`

[5] `https://github.com/RobertInjac/Master-thesis/blob/master/features/non_classic_features.py`

3.2.3. Feature Analysis

The extraction of features creates a lot of potential for examination. The dataset text is hard to analyze numerically or visually, but the features – which are numeric values – are perfect for analysis.

The first thing to investigate are the relations between features. A common way of examining this is by using a correlation matrix. The **correlation matrix** shows the correlation coefficient between each pair of features. Considering no assumption on the distribution of features can be made, the Spearman's correlation coefficient (which is non-parametric) is used (Spearman, 1987).

The correlation matrix is shown in Figure 3.2. Considering $correlation(a, b) = correlation(b, a)$, only the lower triangle of the matrix is shown. A large number of features are highly correlated ($coefficient > 0.5$). Classical features are more often highly correlated with other classical features, and the same goes for non-classical features.

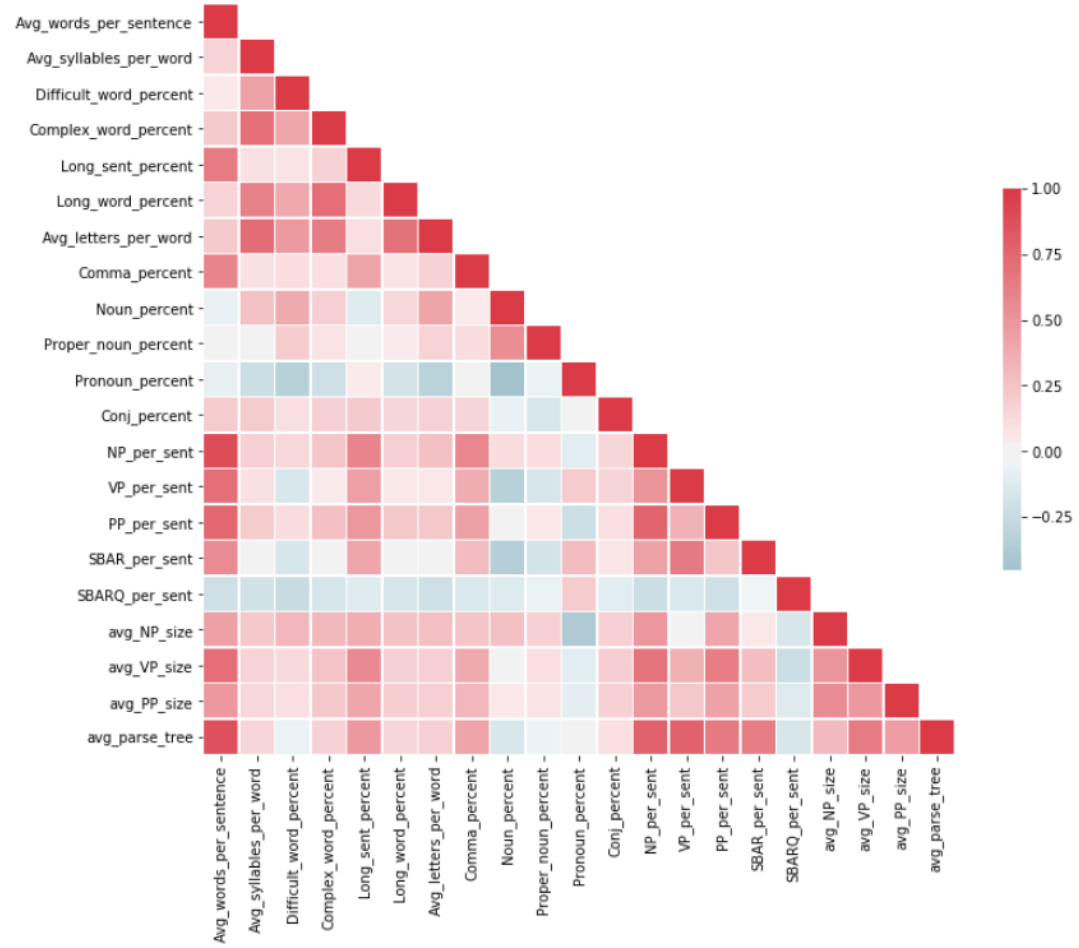

Figure 3.2: Feature correlation matrix

Another interesting thing to investigate are the relations between each feature and the readability level. The readability level should be dependent on the features in some way; otherwise, the features would be useless. This dependence can be examined by **calculating Spearman's correlation coefficient between each feature and the readability level.**[6] The result of these calculations is shown in Table 3.2. It can be seen that most features have a medium amount of correlation with the readability level. Features from both families seem to have members with high and low correlation.

Family	Tag	Spearman correlation	p-value
Classical	Long_sent_percent	0.406835	$1.834\,031 \times 10^{-116}$
Classical	Conj_percent	0.376528	$1.041\,405 \times 10^{-98}$
Non-classical	PP_per_sent	0.345733	$1.647\,374 \times 10^{-82}$
Classical	Avg_words_per_sentence	0.340791	$4.490\,312 \times 10^{-80}$
Classical	Complex_word_percent	0.335279	$2.075\,808 \times 10^{-77}$
Non-classical	avg_VP_size	0.330557	$3.604\,455 \times 10^{-75}$
Classical	Long_word_percent	0.317244	$4.584\,006 \times 10^{-69}$
Classical	Difficult_word_percent	0.307724	$6.909\,395 \times 10^{-65}$
Non-classical	avg_NP_size	0.298653	$4.769\,165 \times 10^{-61}$
Classical	Avg_syllables_per_word	0.286907	$2.806\,479 \times 10^{-56}$
Non-classical	VP_per_sent	0.271278	$2.812\,078 \times 10^{-50}$
Non-classical	NP_per_sent	0.263428	$2.076\,229 \times 10^{-47}$
Non-classical	avg_parse_tree	0.243033	$2.110\,427 \times 10^{-40}$
Non-classical	avg_PP_size	0.235437	$5.932\,760 \times 10^{-38}$
Non-classical	SBAR_per_sent	0.205363	$4.320\,323 \times 10^{-29}$
Classical	Avg_letters_per_word	0.150547	$3.196\,957 \times 10^{-16}$
Classical	Comma_percent	0.147277	$1.390\,681 \times 10^{-15}$
Classical	Pronoun_percent	0.046963	$0.011\,271$
Classical	Proper_noun_percent	-0.201195	$5.767\,141 \times 10^{-28}$
Classical	Noun_percent	-0.245153	$4.222\,745 \times 10^{-41}$
Non-classical	SBARQ_per_sent	-0.253452	$6.672\,200 \times 10^{-44}$

Table 3.2: Correlation of features with the readability level, sorted by the correlation coefficient. The p-value roughly indicates the probability of an uncorrelated system producing datasets that have a Spearman correlation at least as extreme as the one computed from these datasets.

[6] Like before, Spearman correlation is used since no assumption on the distribution of features nor the readability level can be made.

It must be noted that Spearman's coefficient can only capture monotonic relationships; it is possible that there is another kind of dependence between these values. **By visualizing the relationship between two variables it is possible to get a more accurate picture.** One such visualization of the relationship between a feature (*average number of words per sentence*) and the readability level is shown in Figure 3.3. This figure shows a **worrying discovery**. As said before (in the section WeeBit Dataset), the WeeBit dataset is created by combining two corpora – *WeeklyReader* and *BBC-Bitesize*. The readability levels 0, 1, and 2 are from *WeeklyReader* while level 3 and 4 are from *BBC-Bitesize*. The division in the dataset is clearly seen in the figure. It must be noted that not all features show this division. Visualizations for all other features are available online.[7]

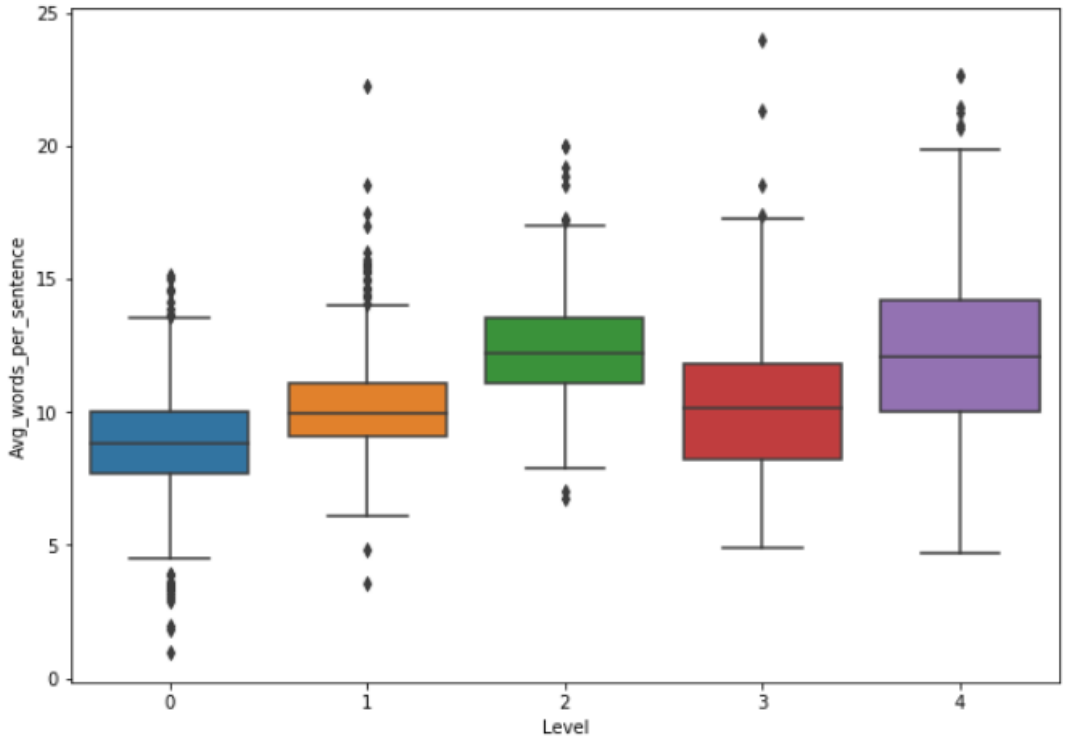

Figure 3.3: Distribution of the feature *average number of words per sentence* for each readability level

[7] https://github.com/RobertInjac/Master-thesis/blob/master/features/feature_analysis.ipynb

4. Readability Formulas

A readability formula is a **mathematical equation that strives to relate the comprehension of the reader and the linguistic characteristics of the text** (Danielson, 1987). Readability formulas take as input some features of the text (for example, *average number of words per sentence*) and output a readability score. Even though the formulas are simple and usually use just a few features, they have been proved to be very effective in the fields of education, business, publishing, law, and medicine (Fry, 2006).

There exists a large number of readability formulas, but some of them are better known and more popular. These popular formulas are based on extensive research and their predictions correlate very well with the results of the actual readability measurements of expert judgments and comprehension tests (Kondru, 2007). This work will use **three formulas always included in the most popular** (Zamanian and Heydari, 2012). These are the **Flesch reading ease**, the **Dale-Chall formula**, and the **Gunning fog index**.

4.1. Flesch Reading Ease

Rudolf Franz Flesch was an Austrian-born American author and readability expert. He was a great proponent of language that is clear and concise, and wrote a book *The Art of Plain Talk* to explain his arguments. In 1948 he developed a readability formula called Flesch reading ease or just Flesch formula (Flesch, 1948). This formula became one of the most popular and heavily tested readability formulas of all time (DuBay, 2007b). The mathematical expression of the formula is shown as expression 4.1. It can be seen that the formula takes two features – *average number of words per sentence* and *average number of syllables per word* – to produce the readability score.

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right) \tag{4.1}$$

Considering the Flesch formula measures reading ease, a higher score represents more readable text. The formula is **very interpretable** – the complete interpretation of the Flesch scores is shown in Table 4.1. For a particulary unreadable sentence, the score can be negative; one sentence[1] from *Moby Dick* has a Flesch score of -146.77.

Flesch Score	Description of sytle	Typical Magazine	School Grade
$90.0 - 100.0$	Very Easy	Comics	4th grade
$80.0 - 90.0$	Easy	Pulp fiction	5th grade
$70.0 - 80.0$	Fairly Easy	Slick fiction	6th grade
$60.0 - 70.0$	Standard	Digests	7th or 8th grades
$50.0 - 60.0$	Fairly Difficult	Quality	Some high school
$30.0 - 50.0$	Difficult	Academic	High school or some college
$0.0 - 30.0$	Very Difficult	Scientific	College

Table 4.1: Flesch score interpretation. Taken from (DuBay, 2007a).

4.2. Dale-Chall Formula

Inspired by Flesch's work, Edgar Dale and Jeanne Chall created their own formula in their 1948 work (Dale and Chall, 1948). Their change was to **count the number of difficult words**. A word is marked as difficult if it is not present in the list of 3000 easy words. The list was created by the authors: the words in the list are words which 80% of fourth-grade students are familiar with. It is available online.[2]

[1] *Though amid all the smoking horror and diabolism of a sea-fight, sharks will be seen longingly gazing up to the ship's decks, like hungry dogs round a table where red meat is being carved, ready to bolt down every killed man that is tossed to them; and though, while the valiant butchers over the deck-table are thus cannibally carving each other's live meat with carving-knives all gilded and tasselled, the sharks, also, with their jewel-hilted mouths, are quarrelsomely carving away under the table at the dead meat; and though, were you to turn the whole affair upside down, it would still be pretty much the same thing, that is to say, a shocking sharkish business enough for all parties; and though sharks also are the invariable outriders of all slave ships crossing the Atlantic, systematically trotting alongside, to be handy in case a parcel is to be carried anywhere, or a dead slave to be decently buried; and though one or two other like instances might be set down, touching the set terms, places, and occasions, when sharks do most socially congregate, and most hilariously feast; yet is there no conceivable time or occasion when you will find them in such countless numbers, and in gayer or more jovial spirits, than around a dead sperm whale, moored by night to a whaleship at sea.* (Melville, 1892)

[2] `https://github.com/RobertInjac/Master-thesis/blob/master/features/resources/dale_chall_easy_word_list.txt`

The Dale-Chall formula is **calculated using the following algorithm**:

1. Calculate score:

$$0.1579 \left[\left(\frac{\text{difficult words}}{\text{total words}} \right) \times 100 \right] + 0.0496 \left(\frac{\text{total words}}{\text{total sentences}} \right)$$

2. If the percentage of difficult words is over 5%, add 3.6365 to the score

Unlike Flesch, a low Dale-Chall score means the text is more readable. This score of this formula is also interpretable – the interpretation is given in Table 4.2.

Dale-Chall Score	Notes
4.9 or lower	easily understood by an average 4th-grade student or lower
$5.0 - 5.9$	easily understood by an average 5th or 6th-grade student
$6.0 - 6.9$	easily understood by an average 7th or 8th-grade student
$7.0 - 7.9$	easily understood by an average 9th or 10th-grade student
$8.0 - 8.0$	easily understood by an average 11th or 12th-grade student
$9.0 - 9.9$	easily understood by an average 13th to 15th-grade (college) student

Table 4.2: Dale-Chall score interpretation. Taken from (Dale and Chall, 1948)

4.3. Gunning Fog Index

Robert Gunning founded the first readability consulting firm dedicated to reducing the *fog* in newspapers and business writing in 1944 (DuBay, 2007b). Almost a decade later, in 1952, he published his Fog index (Gunning, 1952). The index has two features – *average number of words per sentence* and ***percentage of complex words***. A word is complex if it has more than two syllables. The full formula is given as expression 4.2.

$$0.4 \left[\left(\frac{\text{total words}}{\text{total sentences}} \right) + 100 \left(\frac{\text{complex words}}{\text{total words}} \right) \right] \tag{4.2}$$

The Fog index was shown to be a good sign of hard-to-read text (Seely, 2013), although it has its limits. Not all complex words are hard to read. A word such as *interesting* is not generally thought to be a difficult word, but it has four syllables. On the other hand, a short word can be difficult if it is not used by most people on a regular basis.

Like two formulas before, this formula is also interpretable. The interpretation of the Fog index is given in Table 4.3.

Gunning Fog Index	Reading Level by Grade
6	6th grade
7	7th grade
8	8th grade
9	High school freshman
10	High school sophomore
11	High school junior
12	High school senior
13	College freshman
14	College sophomore
15	College junior
16	College senior
17	College graduate

Table 4.3: Gunning fog index interpretation. Taken from (Gunning, 1952)

4.4. Formula Analysis

The readability scores of all three formulas were calculated on the texts of the WeeBit dataset. The first thing to investigate are the relations between each formula's readability score and the readability level. This relationship is examined by **calculating Spearman's correlation coefficient between formula scores and the readability level.** The results are given in Table 4.4. As can be seen, the Gunning fog index correlation value is a bit higher than the rest.[3] The capture the full relationship between formula scores and readability level, their relationship was visualized in Figure 4.1.

The code for implementation and analysis of formulas is available online.[4]

Formula	Spearman correlation	p-value
Flesch reading ease	-0.361127	$2.199\,177 \times 10^{-90}$
Dale-Chall formula	0.349714	$1.667\,135 \times 10^{-84}$
Gunning fog index	0.431514	$2.391\,690 \times 10^{-132}$

Table 4.4: Correlation of formula scores with the readability level. The p-value roughly indicates the probability of an uncorrelated system producing datasets that have a Spearman correlation at least as extreme as the one computed from these datasets.

[3]The full statistical comparison of formulas will be done later in this work.

[4]https://github.com/RobertInjac/Master-thesis/tree/master/formulas

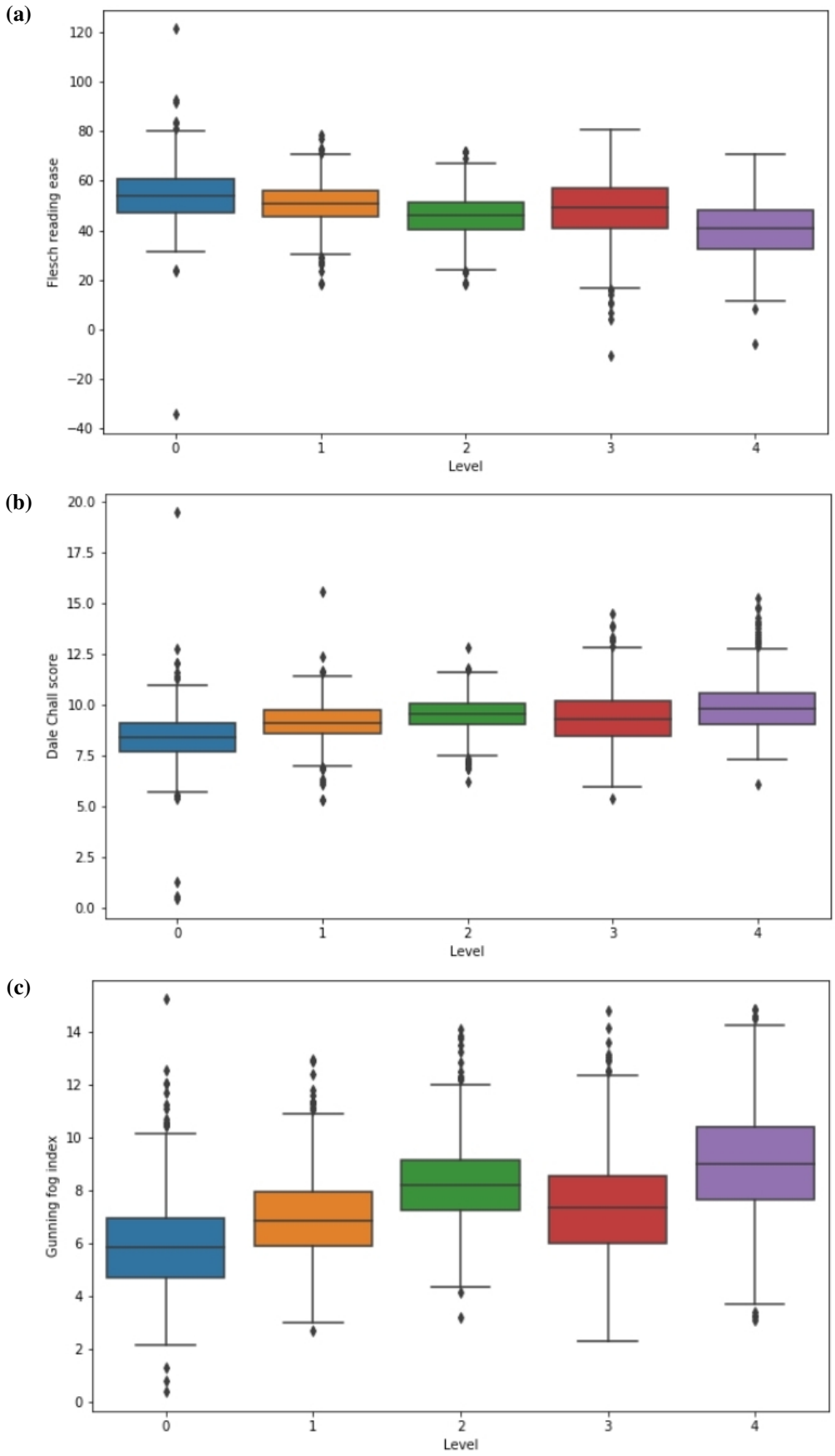

Figure 4.1: Distribution of Flesch (a), Dale-Chall (b), and Gunning fog (c) formula scores for each readability level.

5. Machine Learning Models

Machine learning is generally defined as a field concerned with **programming computers to use example data or past experience to solve a given problem** (Alpaydin, 2009). The fact that a computer system is able solve a problem by *learning* from data (relying on patterns and inference) and not by using pre-defined rules is the key idea in machine learning, giving the field its name. It is generally regarded as a subset of artificial intelligence (AI).

A mathematical model based on sample data (known as *training data*) built by a machine learning algorithm in order to make predictions is called a **machine learning model.** The goal of a machine learning model is not merely to memorize given data. Its core objective is to generalize from its experience (Bishop, 2006). **Generalization** refers to the ability of a model to make accurate predictions on new, unseen examples (known as *test data*) after having experienced the training data. The examples in the training data are sampled from some unknown probability distribution and the machine learning algorithm has learn this distribution in some way to produce accurate predictions for examples in the test data, which also come from the unknown distribution.

The **history of machine learning** begins in the 1940s. The following overview was made according to (Russell and Norvig, 2016). As a scientific discipline, machine learning grew out of the pursuit for artificial intelligence. The AI field was divided into two camps: the researchers who used symbolic methods based on logic, and researchers who built models based on probability and statistics. Several such early machine learning models were invented. However, they had theoretical and practical difficulties, and had fallen out of favor in the 1970s. In 1980s, symbolic methods like expert systems were dominating the field, and machine learning research in AI was practically abandoned.

Machine learning, reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to dealing with solvable problems of practical nature. It also benefited from the increasing size of digital data which can be used for training the models.

Machine learning is divided into several broad tasks:

1. **Supervised learning**

   The task of learning a function that maps an input to an output based on example input-output pairs (Russell and Norvig, 2016). The input is consisted of one or several features, while the output is usually called a target variable/variables. Depending on type of the target variable, there are three types of supervised learning:

   (a) **Classification**

   The target variable is **categorical**. Given the input data, the goal is to predict what is the category/class. The model implementing classification is called a classifier. For example, given an newspaper article, a classifier will decide whether the topic is news, sport or entertainment.

   (b) **Regression**

   The target variable is **numerical**. The model implementing regression is called a regressor. For example, given the time of the day and the city neighbourhood, the regressor will predict electricity consumption.

   (c) **Ordinal regression** (also known as ordinal classification)

   The target variable is **ordinal**. An ordinal variable is a variable whose value exists on an arbitrary scale where only the relative ordering between different values is significant. Example of an ordinal variable is a movie rating from 1 (*very bad*) to 5 (*excellent*). Ordinal regression can be though of as an intermediate problem between regression and classification.

2. **Unsupervised learning**

   The task of drawing inferences from data consisting of input data without labeled responses. The most used method in unsupervised learning is clustering – grouping a set of objects in such a way that objects in the same cluster are more similar (in some sense) to each other than to those in other clusters.

3. **Reinforcement learning**

   The task of learning which actions should software agents perform in an environment so as to maximize some notion of a reward. The difference from supervised learning is that labelled input/output pairs need not be presented – they are observed when performing actions in the environment.

5.1. Readability Prediction using Machine Learning

Machine learning can be used to solve the task of readability prediction. In this work, **the task is to predict the readability level of a text from the WeeBit dataset**. The full solution pipeline is shown in Figure 5.1. The starting point is a text from the WeeBit dataset. Using feature extraction based on natural language processing methods (explained in Chapter 3) the text is converted into a set of features. The features are used as input of the machine learning model. The model outputs an integer from 0 to 4, which represents the readability level.

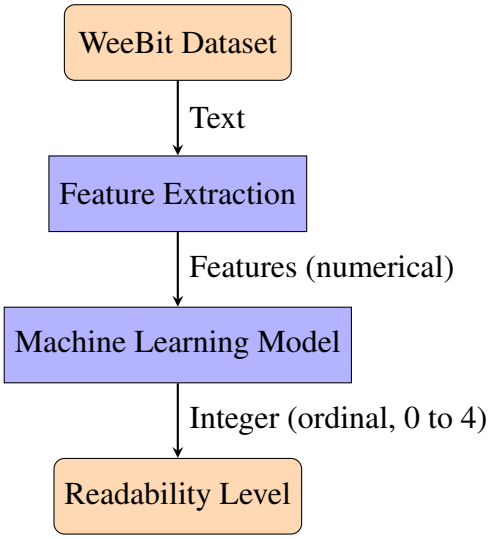

Figure 5.1: Readability prediction pipeline (using machine learning)

Considering the output of the model is an ordinal variable,[1] **this machine learning task is a task of ordinal regression**. Considering ordinal regression is *in-between* regression and classification, it can be solved using both of those methods. The problem when using classification for ordinal regression is the fact that classification algorithms assume classes are categorical variables. Therefore, a classification evaluation measure such as accuracy will count the error of predicting level 3 instead of level 4 as exactly the same as predicting level 0 instead of level 4. On the other hand, the problem with regression is the assumption that the difference between adjacent levels is equal (in other words, that the difference between level 0 and level 1 is the same as the difference between level 1 and level 2), which we have no way to assume.

While a lot of papers use classification measures (such as accuracy) for ordinal

[1] Readability level has an arbitrary scale where only the relative ordering between different readability levels is significant.

regression, is has been shown that using regression measures (such as mean squared error or correlation) gives better results (Gaudette and Japkowicz, 2009). In this work, **Spearman correlation will be used as an evaluation measure**. It is a robust, non-parametric measure well-suited for the task. The Spearman correlation is used in a following way: for each text, its readability level is predicted. This gives a vector of readability levels (each point of the vector is the predicted redability level for that text). The Spearman correlation is calculated between this vector of predicted readability levels and the vector of true readability levels.

5.2. Model Selection

The most important step in solving the task of readability prediction using machine learning is **choosing a machine learning model** (model selection). Considering Spearman correlation is being used as an evaluation measure, it is natural to use regression models.[2] There is a great number of regression models that have been invented. In this work, the focus will be on **four models**:

1. Random forest model,
2. Gradient boosting model,
3. Support vector machine model,
4. Multilayer perceptron.

The choice to use these four models was made after careful consideration. Each of the models has enough complexity to tackle the problem of readability prediction, but is not too complex for that task. A model which is too simple will not be able to learn the underlying structure of the data. In machine learning terms, it would be an **underfitted** model. Its performance on both the train and the test set will be inadequate. On the other hand, a model that is too complex will learn the train data too closely (including the noise present in that data). In machine learning terms, it would be an **overfitted** model. Its performance on the train set will be excellent (maybe even perfect), but it will be unefficient on the test set. An example of the danger of overfitting can be seen in Figure 5.2.

[2]It is also possible to use classification models. However, classification models optimize a classification measure, but the evaluation measure used (Spearman correlation) is a regression measure. Therefore, they are not expected to be as efficient as the regression models.

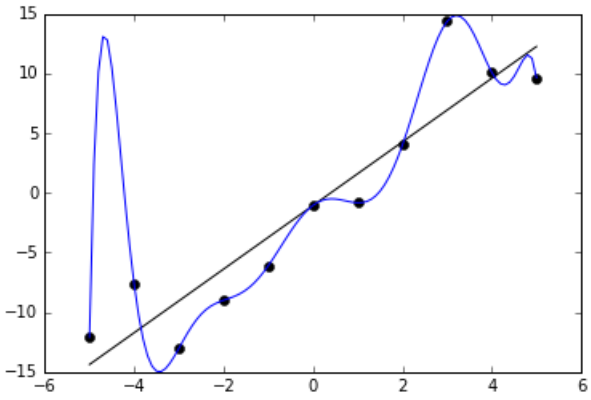

Figure 5.2: Noisy (roughly linear) data is fitted to a linear function and a polynomial function. Although the polynomial function is a perfect fit, the linear function can be expected to generalize better. Taken from (Ghiles, 2016)

In the next four subsections, each of the machine learning models will be explained. After that, their implementations will be described. The full statistical comparison of models, and the comparison of models and formulas will be done in the final chapter.

5.2.1. Random Forest

Before describing what is a random forest model, three key terms need to be explained: decision trees, ensemble learning, and bagging. A **decision tree** is a flowchart-like structure in which each internal node represents a test on a feature, each leaf node represents an output, and branches represent conjunctions of features that lead to those outputs (Yadav, 2018). An example of a decision tree for readability prediction can be seen in Figure 5.3. In machine learning, **ensemble learning** methods use multiple learning models to obtain better predictive performance than could be obtained from any of the constituent learning models alone (Opitz and Maclin, 1999). An example of an ensemble learner is shown in Figure 5.4.

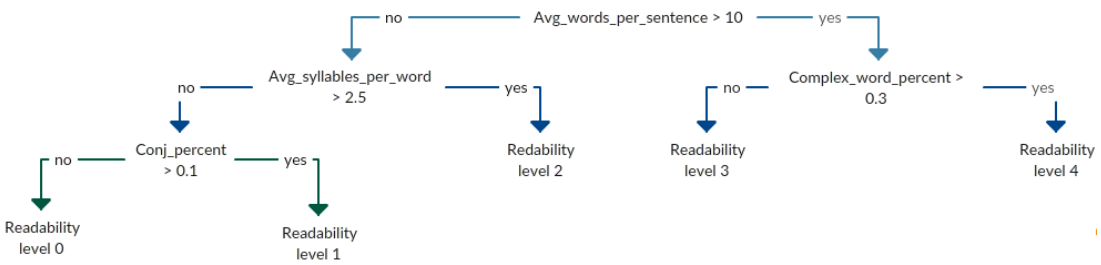

Figure 5.3: A simple decision tree for readability prediction.

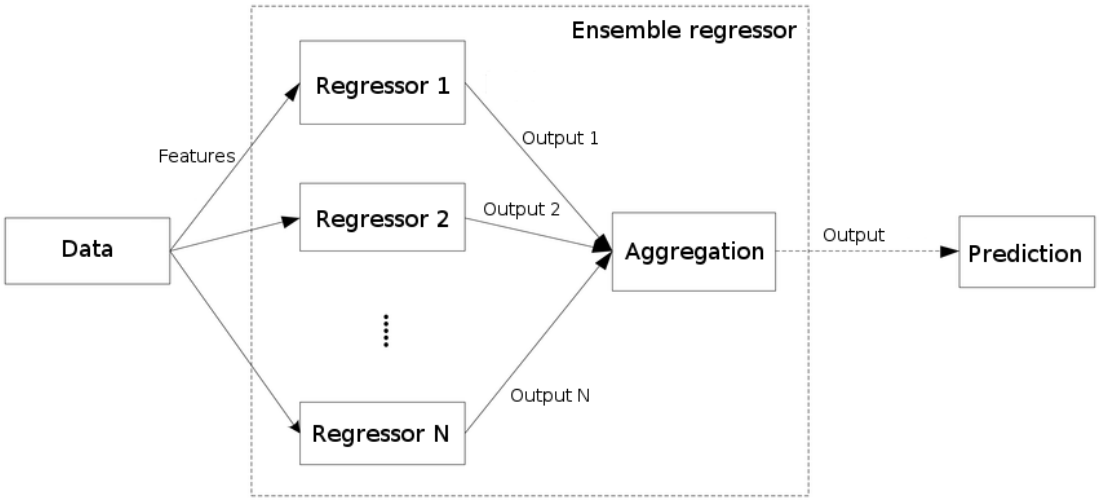

Figure 5.4: Ensemble learning regressor. Taken from (Xu et al., 2015), modified by the author.

Bootstrap aggregating, usually shortened to **bagging**, is an ensemble learning technique designed to prevent overfitting[3] of machine learning models (Friedman et al., 2001). The steps of the method are the following:

1. Given a standard training set D of size n, bagging generates m new training sets D_i, each of size n, by sampling from D uniformly and with replacement. This kind of sample is known as a **bootstrap** sample.[4]

2. Using the m bootstraped training sets, m models are fitted and aggregated. In the case of regression, this is done by averaging the output.

Finally, the definition of a random forest model can be given. Random forest (also called random decision forest) is an ensemble learning method for classification, regression and other tasks that operates by **constructing a multitude of decision trees** (Ho, 1995). The random forest learning algorithm uses bagging (with decision trees being the base models) with one important improvement: each tree gets a bootstrap sample of the features. This is called **feature bagging**: it causes base models to not over-focus on features that appear highly predictive in the training set, but fail to be as predictive for examples outside that set. The full architecture of the random tree model is shown in Figure 5.5.

[3]In statistical terminology, bagging is a technique for reducing the variance of a statistical model. The model with a high-variance is just another name for an overfitted model. The term *high-variance* is mostly used in statistics, while *overfitting* is mostly used in machine learning.

[4]Therefore, the procedure is called *bootstrap* aggregating.

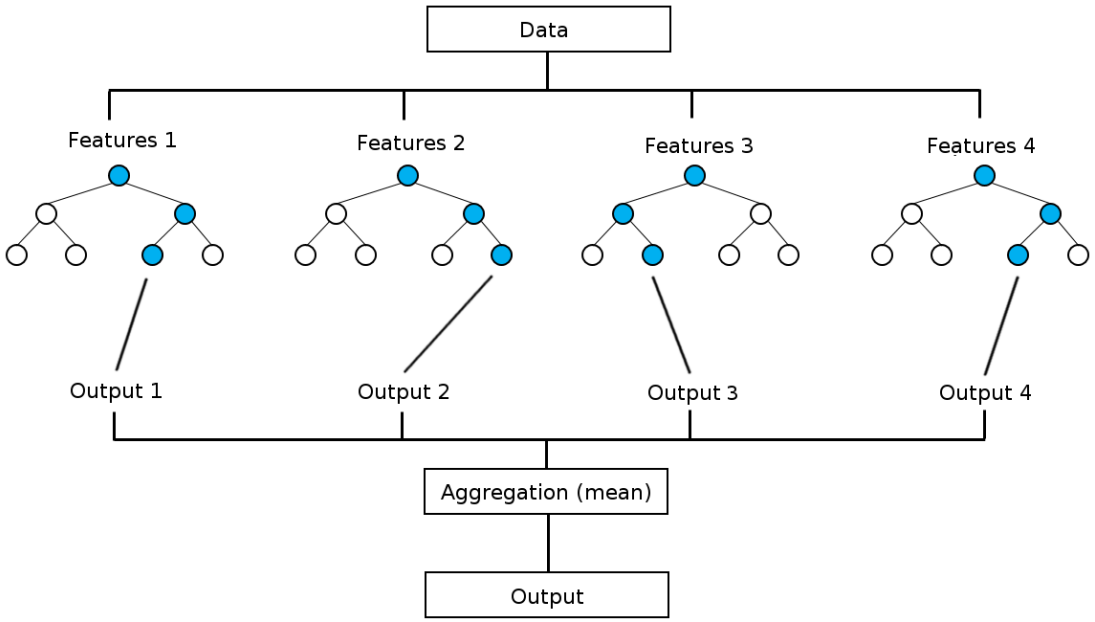

Figure 5.5: Random forest model architecture. Taken from (*Random Forest Classifer*, 2018), modified by the author.

These methods (bagging and feature bagging) are the reason why a random forest models outperforms a decision tree model (which is much simpler). Using bagging and feature bagging, it **corrects the decision trees' habit of overfitting to their training set** (Friedman et al., 2001). As a consequence, random forest models are very popular, and are implemented in a variety of machine learning libraries and packages.

5.2.2. Gradient Boosting

The gradient boosting model is similar to random forest in the sense that it is also an ensemble learning method which constructs a multitude of decision trees. However, instead of bagging, the gradient boosting algorithm is based on boosting. **Boosting** is a procedure based on the idea that the outputs of many *weak* base models are combined to produce a powerful model (Friedman et al., 2001). Unlike bagging where each base model is trained independently and then their outputs are aggregated without preference to any model, boosting relies on a sort of *teamwork* between base models. **Each base model will *focus* more on the examples on whom the previous model failed.** The way to achieve the *focusing* differs depending on the boosting algorithm used. A simple way to achieve this is by giving more weight to the wrongly predicted examples. An example of boosting using this method is shown in Figure 5.6.

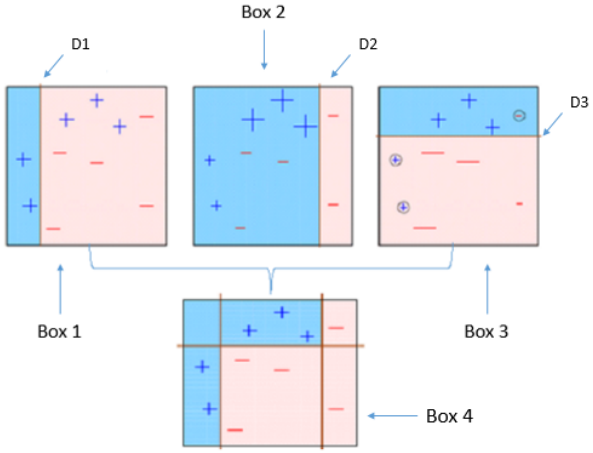

Figure 5.6: Boosting used to solve a classification problem. In each step, more weight is given to the misclassified examples of the previous step. Taken from (Sunil, 2018)

The gradient boosting method achieves the *focusing* by performing a **minimization of an arbitrary differentiable loss function**. Each new model gradually minimizes the loss function of the whole system using gradient descent.[5] Without going into mathematical details, the intuition behind this is shown and explained in Figure 5.7. Gradient boosting is a very powerful machine learning model, and it has shown considerable success in a wide range of practical applications (Friedman et al., 2001).

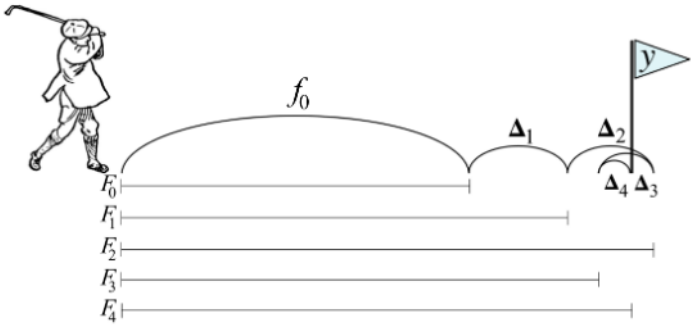

Figure 5.7: Intuition behind gradient boosting. It is helpful to think of gradient boosting approach as a golfer initially whacking a golf ball towards the hole at y but only getting as far as f_0. The golfer then repeatedly taps the ball more softly, working the ball towards the hole, after reassessing the distance to the hole at each stage. Taken from (Parr and Howard, 2018)

[5]Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point (Boyd and Vandenberghe, 2004).

5.2.3. Support Vector Machine

Support-vector machine (SVM) is a supervised learning model that can be used for classification or regression. The SVM model is much easier to explain on the example of classification. The following description is based on (Cristianini et al., 2000). In the classification case, the idea of an SVM is to find a hyperplane in an N-dimensional space (N being the number of features) that distinctly classifies the data points. Specifically, the idea is not to find any hyperplane, but the hyperplane with the **maximum margin**, with margin being the largest distance to the nearest training example of any class. The maximum margin is completely determined by the examples closest to it – these are called **support vectors**, giving the model its name. An example of the maximum margin and its support vectors is shown in Figure 5.8. Intuitively, the larger the margin, the generalization error of the classifier should be lower.

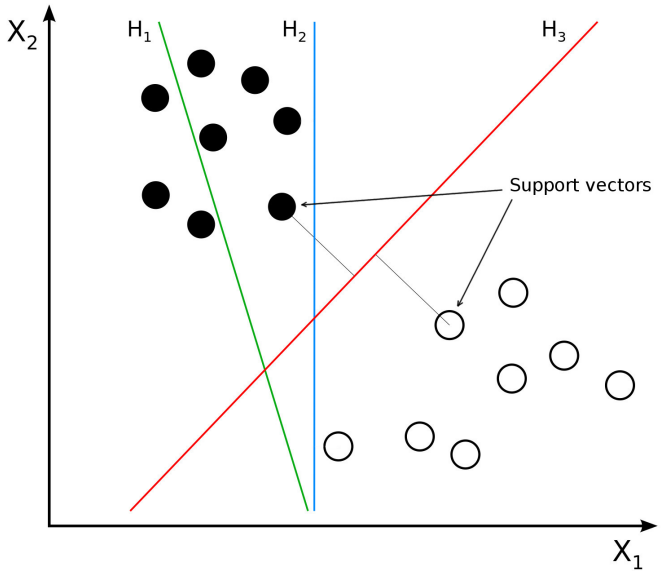

Figure 5.8: Maximum margin (H_3) and its support vectors. Hyperplane H_1 does not correctly classify the data. Hyperplane H_2 does correctly classify the data, but is not the maximum margin. Taken from (Weinberg, 2012), modifed by the author.

The SVM model in its original form is a linear classifier. It cannot classify non-linearly separable data. However, a method called **kernel trick** was invented for this problem (Boser et al., 1992). The idea behind it is that our data, which is not linearly separable in some n-dimensional space, may be linearly separable in a higher dimensional space. The mapping into a higher dimensional space is done using a kernel function. An example of the kernel trick is shown in Figure 5.9. Using the trick, the SVM model becomes a powerful non-linear classifier (Jin and Wang, 2012).

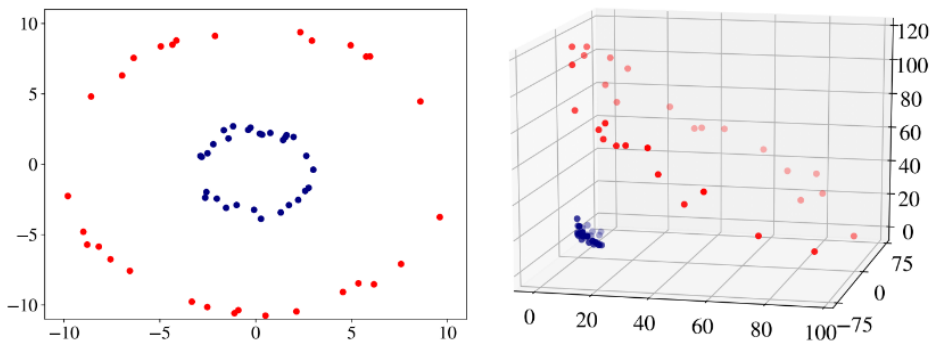

Figure 5.9: Kernel trick example. The data, residing in 2-dimensional space, is non-linearly separable. Using a kernel function, the data is transferred into 3-dimensional space. In 3-dimensional space, the data is linearly separable. Taken from (Burkov, 2019).

For a long time, SVM was used just for classification. In 1996, a version of the SVM for regression, called **support-vector regression** (SVR) was proposed (Drucker et al., 1997). The SVR is similar to SVM with a few differences. The general idea is to fit the regression error within a certain threshold. The kernel trick works the same way for regresson as for classification, and allows for non-linear regression.

5.2.4. Multilayer Perceptron

Before explaining multilayer perceptrons, the concept of perceptron needs to be defined. **Perceptron** is a machine learning model based on the human neuron. The architecture of the model is shown in Figure 5.10. It takes several inputs and runs their weighted sum through an activation function. Perceptrons are capable of learning linearly separable patterns (Rojas, 2013). However, they are not capable of learing non-linear patterns. In a famous example, it was shown that perceptrons are unable to learn the XOR function (Minsky and Papert, 2017).

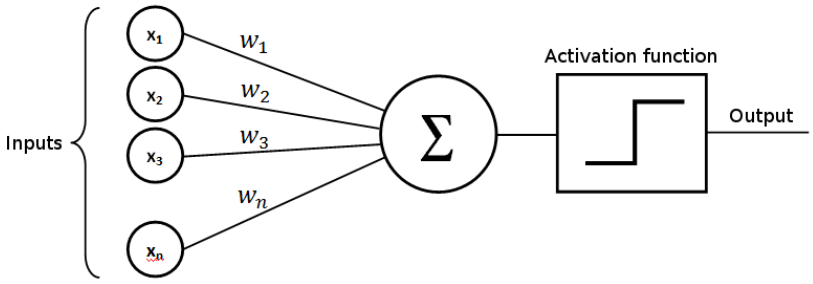

Figure 5.10: Architecture of the perceptron. Taken from (Mayranna, 2013), modified by the author.

The perceptrons, however, can be combined. The output of one perceptron can be used as an input to another perceptron. In this way, they can be arranged into a network. This network is called a **multilayer perceptron** (MLP), and it is the most simple example of a neural network.[6] A MLP needs to consist of at least three layers of perceptron nodes: an input layer, a hidden layer and an output layer. The nodes of the hidden layers need to use a nonlinear activation function. Due to that nonlinarity, **MLPs can learn nonlinar patterns** (Cybenko, 1989). MLPs can be used for both classification and regression – the architecture can stay the same, only the output node's activation function needs to be changed. An example of a MLP model is shown in Figure 5.11.

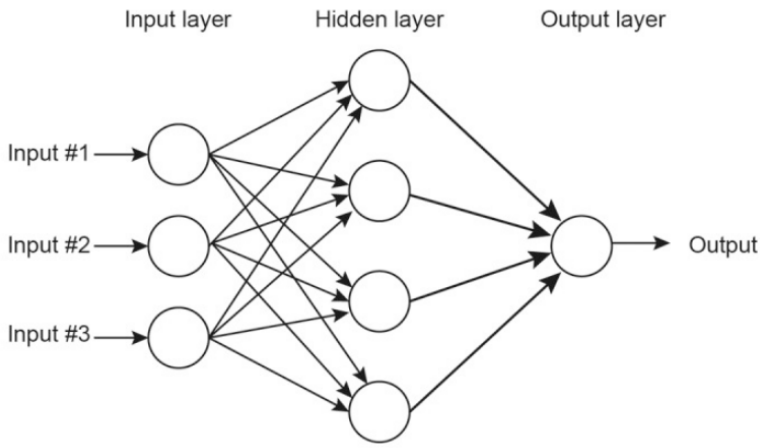

Figure 5.11: MLP with three input nodes, one output node, and one hidden layer consisting out of 4 nodes. Taken from (Manning et al., 2014).

To learn the weights of each node, an algorithm called backpropagation is used. **Backpropagation** is an iterative, recursive and efficient method for calculating the weights updates to improve the network until it is able to perform the task for which it is being trained (Goodfellow et al., 2016). The algorithm uses a gradient descent approach that exploits the chain rule to propagate the error from the output through the hidden layers of the network.

MLP is a very powerful model. Soon after its invention, it has found its use in areas such as speech recognition, image recognition, and machine translation software (Wasserman and Schwartz, 1988). With invention of recurrent neural networks (RNNs), which can use their internal state (memory) to process sequences of inputs (such as words in a sentence), their use has fallen in the NLP domain. However, they still re-

[6]MLPs are colloquially referred to as *vanilla* neural networks, especially when they have a single hidden layer (Friedman et al., 2001)

main competetive in tasks not involving sequences. The task of readability prediction in this work, where features have been extracted from the text, is one such task.

5.3. Model Implementation and Analysis

All four models have been implemented in Python and evaluated on the WeeBit dataset. The details of each implementation will be described below. As said before, the metric used for evaluation was Spearman correlation.

The **random forest model** was implemented using the *scikit-learn* implementation in Python.[7] This model is a regressor, but the goal is to find the readability level, which is an ordinal variable. To do this, the output of the regressor model was discretized in a following way: it was rounded, with all values below 0.5 given readability level of 0, and all values above 3.5 given level of 4. The model has two **hyperparemeters**:[8] number of estimators and maximum depth of a tree. To find the best hyperparameters, grid search[9] was used. The best hyperparemeters were found to be 20 for the maximum depth, and 100 for the number of estimators. Using this hyperparameters, the model was trained. The implementation of the model is available online.[10]

The **gradient boosting model** was implemented using the Python API of the XGBoost framework.[11] **XGBoost** (e**X**treme **G**radient **Boost**ing) is a scalable, portable, and distributed gradient boosting library. The model provided by XGBoost (XGBRegressor) is a regressor, so the dicretization of its output was performed in the same way as for the random forest model. The model has two hyperparameters, the same ones as random forest: number of estimators and maximum depth of a tree. A grid search was performed and the best hyperparameters were found: 5 for the maximum depth, and 200 for the number of estimators. Using this hyperparameters, the model was trained. The implementation of the model is available online.[12]

[7] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html`

[8] In machine learning, a *hyperparameter* is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training.

[9] Grid search is simply an exhaustive search through a manually specified subset of the hyperparameter space.

[10] `https://github.com/RobertInjac/Master-thesis/blob/master/ml_models/models/random_forest.py`

[11] `https://github.com/dmlc/xgboost`

[12] `https://github.com/RobertInjac/Master-thesis/blob/master/ml_models/models/xgboost.py`

The **support vector machine model** was implemented using the *scikit-learn* implementation in Python.[13] The model is a regressor, and the discretization of its output was done in the same way as the previous two models. Linear kernel was used. The model has one hyperparameter – the regularization parameter C. Using a predefined range of possible Cs, a search was performed. The best parameter C was found to be 20.0. Using this hyperparameter, the model was trained. The implementation of the model is available online.[14]

The **multilayer perceptron model** was implemented using the Keras[15] framework with TensorFlow[16] back-end. Keras is an open-source neural-network library written in Python, while TensorFlow is a free and open-source software library for differentiable programming across a range of tasks, mostly used for machine learning. The details of the network architecture used are shown in Figure 5.12. This architecture was chosen using an informal process: a couple of architectures were tried, and the one performing best on the test set data was chosen. The optimizer used to train the model is the ADAM optimizer, a variation of the gradient descent algorithm. The model is a regressor, and the discretization of its output was done in the same was as the previous three models. The implementation of the model is available online.[17]

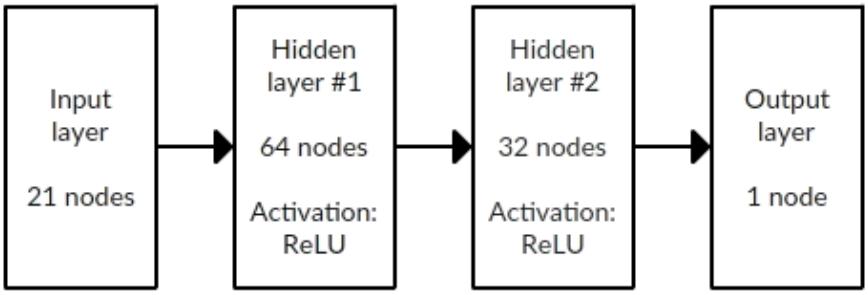

Figure 5.12: Architecture of the MLP model used in this work. The input size (21) corresponds with the 21 extracted features. The output is one number – the readability level. Rectified linear unit (ReLU) is a popular non-linear activation function used in deep neural networks.

[13] https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

[14] https://github.com/RobertInjac/Master-thesis/blob/master/ml_models/models/support_vector_machine.py

[15] https://keras.io/

[16] https://www.tensorflow.org/

[17] https://github.com/RobertInjac/Master-thesis/blob/master/ml_models/models/multilayer_perceptron.py

After implementation and training, the models were evaluated on the WeeBit test set. The results of the evaluation are available in Table 5.1. As can be seen, the **MLP model achieved the best Spearman correlation value**. The random forest model is quite close. Code used for the evaluation of models is available online.[18] The full statistical comparison of models will be done in the next chapter.

Model	**Spearman correlation**	R^2 **score**
Random Forest	0.776516	**0.590388**
Gradient Boosting	0.731930	0.523064
Support Vector Machine	0.758866	0.547868
Multilayer perceptron	**0.785263**	0.578341

Table 5.1: Evaluation of machine learning models. Along with the Spearman correlation, the R^2 score is shown. Both are popular evaluation metrics for regression.

[18] `https://github.com/RobertInjac/Master-thesis/blob/master/ml_models/model_evaluation.ipynb`

6. Comparison

In previous chapters, the task of readability prediction was solved using various formulas and machine learning models. In this chapter, **the goal is to compare them and find out which one is the best**. This could be done by just comparing the Spearman correlation values which were reported in Tables 4.4 and 5.1. However, these values could be the result of a coincidence. One model has the Spearman correlation value larger than the other model, but using a different train-test split, the situation could be reverse. Therefore, there is a need to use statistical hypothesis testing to find which model is better than others in statistically significant terms.

Statistical hypothesis testing is a method of statistical inference used for comparison of two sets of data. A hypothesis is proposed for the statistical relationship between the two sets of data, and this is compared as an alternative to an idealized null hypothesis that proposes no relationship between the two sets. The comparison is said to be **statistically significant** if the relationship between the sets would be an unlikely realization of the null hypothesis according to a threshold probability – the significance level (Rice, 2006). There are many types of statistical tests. In this work, bootstrap significance testing will be used.

6.1. Bootstrap significance testing

The boostrap significance testing in this work is done according to (Berg-Kirkpatrick et al., 2012), and the following overview is based on that paper.

Before the boostrap part, more details specific to comparison of two models using hypothesis testing must be noted. When comparing model A to model B, the goal is to know if A is better than B on some large population of data. It is found that on some small test set X model A beats B by $\delta(X)$. To know if this difference is statistically significant, it must be known how likely it would be that a new, independent test set X' would show a similar victory for A assuming that A is no better than B on the population as a whole – this assumption is the null hypothesis H_0. Hypothesis testing

consists of attempting to estimate this likelihood, called **p-value**. If the p-value is below the predefined significance level, H_0 can be rejected. In this case, A is better than B in a statistically significant way (in other terms, it was accepted that A's victory was real and not just a random fluke).

In most cases the p-value is not easily computable and must be approximated. **The bootstrap significance testing estimates the p-value though a combination of simulation and approximation.** It draws many simulated test sets X_i and counts how often A sees an accidental advantage of $\delta(X)$ or greater. The tests sets X_i are drawn from X itself, sampling from it with replacement – they are bootstrap samples of X.

The bootstrap significance testing was implemented in Python. There is a function which takes the predicted values of model A and model B on the test set. The number of bootstraps samples to create is also given. In this work, 10^4 samples were used for each evaluation. The estimated p-value is returned. Code of this function is available online.[1]

6.2. Comparison of formulas

The three formulas used in this work are the Flesch formula, the Dale-Chall formula, and the Gunning fog index. Their results on the test set were seen in Table 4.4. The Gunning fog index has the highest Spearman correlation value.

The bootstrap significance testing was performed to see if the Gunning fog index is better than the other two formulas in a statistically significant way. The significance level was set to 0.05. Cosidering the p-values for both tests were much smaller than the significance level, it was accepted that **the Gunning fog index is better than the other two formulas in a statistically significant way.**

6.3. Comparison of machine learning models

The machine learning models used in this work are the random forest model, the gradient boosting model, the support vector machine model and the multilayer perceptron (MLP) model. Their results on the test set were seen in Table 5.1. The random forest model and the MLP model had the highest Spearman correlation value, with MLP performing slightly better than random forest.

[1] `https://github.com/RobertInjac/Master-thesis/blob/master/comparison/bootstrap.py`

The boostrap significance testing was performed to see if the difference between those two models and the rest is statistically significant, and also is the MLP model significantly better than the random forest model. The significance level was set to 0.05. The testing showed that there is **almost no statistically significant difference between different models**. The only thing which was proved is that **random forest model and the MLP model outperform the gradient boosting model in a statistically significant way.**

6.4. Formulas versus models

The only thing left to compare are machine learning models and formulas. Considering the Gunning Fog index performed the best of the formulas, and MLP performed best of the models, those two were compared. Boostrap significance testing was performed to see if the MLP model outperfoms the Gunning fog index. The estimated p-value was very small, much smaller than the chosen significance level (0.05). The null hypothesis was rejected, which means that the **MLP model is better than the Gunning fog formula in statistically significant terms**.

The conclusion is that **machine learning models are truly better than traditional formulas**. Considering they use much more features and are able to learn from them, this comes to no surprise.

7. Conclusion

Readability is an important characteristic of a text. Therefore, the ability to predict readability – known as readability prediction – has a plethora of uses in education, business, publishing, and law. This is a task usually achieved by traditional readability formulas. However, natural language processing techniques, coupled with machine learning, have started to be used in this domain.

The goal of this work was to make a survey of the task of readability prediction, to explain and implement NLP methods which aid in that task, to describe existing readability formulas, to create machine learning models for the task, and to carry a statistical analysis of the results. The survey of the task of readability prediction was done in detail. A plethora of NLP methods were described and implemented. A summary of readability formulas was shown. Four machine learning models were created and implemented. Finally, statistical tests were performed on the formulas and the models. The results show that machine learning models outperform the traditional readability formulas.

There are many ways in which this work could be improved upon. The advances in machine learning such as feature learning can eliminate the feature extraction step. This would require much more data, leading to one more possible improvement – creation of new, large, public-available readability evaluation datasets. Another direction of improvement would be to make more interpretable models which could be used instead of formulas in practice. In any case, there is a lot of room for new approaches to readability with a machine learning perspective.

LITERATURE

Alpaydin, E. (2009). *Introduction to machine learning*. MIT press.

Balakrishna, S. V. (2015). Analyzing text complexity and text simplification: Connecting linguistics, processing and educational applications. *These de doctorat, Eberhard Karls Universität Tübingen, Tübingen, Germany*.

Berg-Kirkpatrick, T., Burkett, D., and Klein, D. (2012). An empirical investigation of statistical significance in nlp. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005. Association for Computational Linguistics.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Burkov, A. (2019). *The Hundred-Page Machine Learning Book*.

Cambria, E. and White, B. (2014). Jumping nlp curves: A review of natural language processing research. *IEEE Computational intelligence magazine*, 9(2):48–57.

Choldin, M. T. (1979). Rubakin, nikolai aleksandrovic. *Encyclopedia of library and information science*, 26:178–179.

Collins-Thompson, K. (2014). Computational assessment of text readability: A survey of current and future research. *ITL-International Journal of Applied Linguistics*, 165(2):97–135.

Cristianini, N., Shawe-Taylor, J., et al. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

Dale, E. and Chall, J. S. (1948). A formula for predicting readability: Instructions. *Educational research bulletin*, pages 37–54.

Danielson, K. E. (1987). Readability formulas: a necessary evil? *Reading Horizons*, 27(3):4.

Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., and Vapnik, V. (1997). Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161.

DuBay, W. H. (2007a). The classic readability studies. *Online Submission*.

DuBay, W. H. (2007b). *Smart Language: Readers, Readability, and the Grading of Text.* ERIC.

Flesch, R. (1948). A new readability yardstick. *Journal of applied psychology*, 32(3):221.

François, T. and Miltsakaki, E. (2012). Do nlp and machine learning improve traditional readability formulas? In *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations*, pages 49–57. Association for Computational Linguistics.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Fry, E. (2006). Readability. *Reading Hall of Fame Book*.

Fry, E. B. (1987). The varied uses of readability measurement today. *Journal of reading*, 30(4):338–343.

Gaudette, L. and Japkowicz, N. (2009). Evaluation methods for ordinal classification. In *Canadian Conference on Artificial Intelligence*, pages 207–210. Springer.

Ghiles (2016). Overfitted data. `https://en.wikipedia.org/wiki/Overfitting#/media/File:Overfitted_Data.png`. [Online; accessed 14-June-2019].

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Gunning, R. (1952). The technique of clear writing.

Ho, T. K. (1995). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Jin, C. and Wang, L. (2012). Dimensionality dependent pac-bayes margin bound. In *Advances in neural information processing systems*, pages 1034–1042.

Kakkonen, T. (2007). *Framework and resources for natural language parser evaluation*. Tuomo Kakkonen.

Kao, A. and Poteet, S. R. (2007). *Natural language processing and text mining*. Springer Science & Business Media.

Kitaev, N. and Klein, D. (2018). Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:1805.01052*.

Kondru, J. (2007). Using part of speech structure of text in the prediction of its readability.

Lively, B. A. and Pressey, S. L. (1923). A method for measuring the vocabulary burden of textbooks. *Educational administration and supervision*, 9(389-398):73.

Manning, C. D., Manning, C. D., and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Manning, T., Sleator, R. D., and Walsh, P. (2014). Biologically inspired intelligent decision making: a commentary on the use of artificial neural networks in bioinformatics. *Bioengineered*, 5(2):80–95.

Mayranna (2013). Perceptron. `https://commons.wikimedia.org/wiki/File:Perceptron_moj.png#/media/File:Perceptron_moj.png`. [Online; accessed 15-June-2019].

McKinney, W. (2010). Data structures for statistical computing in python. In van der Walt, S. and Millman, J., editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56.

Melville, H. (1892). *Moby Dick: or, the white whale*. Page.

Minsky, M. and Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. MIT press.

Nadkarni, P. M., Ohno-Machado, L., and Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198.

Parr, T. and Howard, J. (2018). How to explain gradient boosting. `https://explained.ai/gradient-boosting/index.html`. [Online; accessed 15-June-2019].

Rice, J. A. (2006). *Mathematical statistics and data analysis*. Cengage Learning.

Rojas, R. (2013). *Neural networks: a systematic introduction*. Springer Science & Business Media.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Seely, J. (2013). *Oxford Guide to Effective Writing and Speaking: How to Communicate Clearly*. OUP Oxford.

Sherman, L. A. (1893). *Analytics of literature: A manual for the objective study of English prose and poetry*. Ginn.

Spearman, C. (1987). The proof and measurement of association between two things. *The American journal of psychology*, 100(3/4):441–471.

Sunil, R. (2018). Quick introduction to boosting algorithms in machine learning. `https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/`. [Online; accessed 15-June-2019].

Random Forest Classifer (2018). Random forest classifier. `http://www.globalsoftwaresupport.com/wp-content/uploads/2018/02/ggff5544hh.png`. [Online; accessed 14-June-2019].

Trogkanis, N. and Elkan, C. (2010). Conditional random fields for word hyphenation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 366–374. Association for Computational Linguistics.

Vajjala, S. and Lučić, I. (2018). OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 297–304, New Orleans, Louisiana. Association for Computational Linguistics.

Vajjala, S. and Meurers, D. (2012). 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. Association for Computational Linguistics.

Wasserman, P. D. and Schwartz, T. (1988). Neural networks. ii. what are they and why is everybody so interested in them now? *IEEE Expert*, 3(1):10–15.

Weinberg, Z. (2012). Svm separating hyperplanes. `https://commons.wikimedia.org/wiki/File:Svm_separating_hyperplanes_(SVG).svg`. [Online; accessed 15-June-2019].

Xu, J., Yao, L., and Li, L. (2015). Argumentation based joint learning: a novel ensemble learning approach. *PloS one*, 10(5):e0127281.

Yadav, P. (2018). Decision tree in machine learning. `https://towardsdatascience.com/decision-tree-in-machine-learning-e380942a4c96`. [Online; accessed 14-June-2019].

Zamanian, M. and Heydari, P. (2012). Readability of texts: State of the art. *Theory & Practice in Language Studies*, 2(1).

Procjena čitljivosti teksta postupcima obrade prirodnog jezika

Sažetak

Čitljivost teksta je definirana kao lakoća kojom čitatelj može razumjeti pisani tekst. Formule za čitljivost su tradicionalno bile korištene za određivanje čitljivosti teksta. Posljednih godina, za rješavanje ovog problema počele su se koristiti methode obrade prirodnog jezika zajedno sa strojnim učenjem – ovo je pristup koji je korišten u ovom radu. Metode obrade prirodnog jezika su korištene za ekstrakciju relevantnih značajki iz teksta. Te značajke su dane na ulaz nekoliko modela zasnovanih na strojnom učenju koji predviđaju razinu čitljivosti. Ti modeli su evaluirani na skupu podataka WeeBit. Uspoređeni su jedan s drugim te s tradicionalnim formulama korištenjem statističkih testova. Testovi su pokazali da modeli zasnovani na strojnom učenju postižu bolje rezultate od tradicionalnih formula za čitljivost.

Ključne riječi: čitljivost, procjena čitljivosti, formule za čitljivost, obrada prirodnog jezika, strojno učenje

Predicting Text Readability using Natural Language Processing Methods

Abstract

Text readability is defined as the ease with which a reader can understand a written text. Traditionally, readability formulas have been used to determine the readability of a text. In recent years, natural language processing (NLP) methods coupled with machine learning have been applied to solve this task – this is the approach used in this work. NLP methods have been used to extract relevant features from a text. These features are given as an input to several machine learning models which predict readability level. The models were evaluated on the WeeBit dataset. They have been compared with each other and with traditional formulas using statistical significance testing. The tests showed that machine learning models outperform traditional readability formulas.

Keywords: readability, readability formulas, readability prediction, natural language processing, machine learning