



DEPARTMENT OF PHILOSOPHY,
LINGUISTICS AND THEORY OF SCIENCE

WHEN COMPLEXITY IS NOT A SOLUTION TO THE PROBLEM: THE STUDY OF USING SIMPLE ENGINEERED LANGUAGE FEATURES FOR THE TASK OF TEXT POPULARITY PREDICTION

Hannah Wiltshire

Bachelor's thesis:	15 credits
Course:	Linguistics In-depth course
Level:	First cycle
Semester and year:	Autumn, 2020
Supervisor:	Nikolai Ilinykh
Examiner:	Moa Ekbon
Report number:	(number will be provided by the administrators)
Keywords:	Text classification, feature engineering, text popularity prediction, machine learning

Abstract

This thesis studies the performance of machine learning classifiers trained with linguistic features. The main goals are (1) to identify if manually engineered features could be useful within the field of text popularity prediction and (2) to assist other linguists in their work with automatic tools for data analysis.

A dataset of user-generated comments, consisting of 16840 texts, was scraped from the forum network Reddit and used to train text classifiers which use either (1) Naive Bayes or (2) Support vector machine algorithm. Overall, twenty-four models have been trained. The features are divided into two categories called “standard features” and “engineered features”. Computational linguists commonly use the standard features (such as TF-IDF and bag-of-words) for the text classification tasks. Some examples of the engineered features include the count of significant words and the number indicating the similarity with a topic. We manually constructed such features for Reddit comments for the purpose of this research.

In this thesis, we demonstrate that using engineered features can improve the accuracy of the text classifiers for the task of text popularity prediction.

Acknowledgements

I want to express my gratitude towards my supervisor Nikolai Ilinykh, who guided me through the whole research and writing process. Thank you.

Thank you also to Daniel Leppänen, for your invaluable support with programming.

Thank you, Joe Baxter, for proofreading.

Contents

1. Introduction	1
2. Background	4
2.1. Naive Bayes	4
2.1.1. Formal definition	5
2.1.2. Bag-of-words	8
2.1.3. Laplace smoothing	10
2.1.4. Bag-of-bigrams	11
2.1.5. TF-IDF	12
2.1.6. An example of training Naive Bayes with bag-of-words features	13
2.2. Support Vector Machine (SVM)	16
2.3. Process of machine learning	17
2.3.1. Getting data	17
2.3.2. Feature extraction	17
2.3.3. Training and validating the model	17
2.3.4. Some definitions	18
2.4. Reddit	20
3. Methodology	21
3.1. The dataset	22
3.1.1. Statistics	23
3.1.2. About the scraping tool	26
3.2. Experiments	27
3.2.1. Dataset cut out	28
3.3. Features	30
3.3.1. Definitions	30
3.3.2. SF-sets	31
3.3.3. The EF-set	32
4. Results	35
4.1. Accuracy bar charts	35
4.1.1. Interpretation of the results	38
4.2. Confusion matrices	39
4.3. Sample comments	42
5. Discussion	44
5.1 Future research	48
6. Conclusion	49
References	50

1. Introduction

Nowadays, the amount of information on the Web is vast, and the number of topics that humans discuss online is immense. Social media, forums and blogs are only a few examples of the platforms that we (humans) use today to help us communicate. Thus, the Internet can be described as the most significant source of information. In practice, this information can be utilised in multiple ways. For example, a marketing worker's goal could be to attract human attention to their product advertisement published online. To achieve that, the worker could look at other promotions, determine the features that make humans "like" or "dislike" these texts and try to construct a very well-described, attractive product advertisement. However, while it is generally easy for humans to categorise and interpret the meaning behind a written text, learning to classify text into two or more categories (e.g., likeable / non-likeable) could be a tiresome and time-consuming task. Therefore, an automatic system capable of performing a text classification task could be used for this purpose. However, at the same time, it is not a very simple task for a machine. For example, an automatic system can struggle to capture the pragmatic meaning behind the text or what the text is genuinely saying (Cherpas, 1992).

This research aims to identify which type of text representations (text features) are beneficial for the task of text classification. We seek to determine a path forward for further analysis of the written text, specifically in the task of popularity prediction of the human comments extracted from the online forum Reddit. In this thesis, we focus on the investigation of the effects of using different text features for the task at hand and how these features affect a model's performance.

Reddit is an openly accessible social network with more than 430 million active monthly users (Lin, 2020) and is among the most popular websites in the world (Alexa, n.d.). Various linguistic research projects have used Reddit in the past as a corpus or database for sentiment analysis or other linguistic tasks. For example, in the medical field, Foufi et al. (2019) looked at how persons with chronic diseases expressed themselves and what their relation to user-generated online content was. In this thesis, we are using this vast amount of data for the task of text popularity prediction. We define this task as follows: the text is considered to be popular if it is rated as "likeable" by many humans. At the same time, it is unpopular if it has a fewer number of such "likes" (in the Web terminology, a "like" is an indicator that a human supports the text and probably shared the point of view that text describes). Our goal is to teach an

automatic system to predict whether the comment is popular or not based on different text representations.

The problem of predicting text popularity typically involves the use of the number of complex machine learning methods that learn about the content of the text and make a decision based on this information (Subramanian et al., 2018). Note that here we are looking at the more straightforward and interpretable approach of using text features and feeding them to some simple classification algorithms (Naive Bayes, for example).

We describe **two** main contributions of this thesis:

- **Is it possible to predict text popularity based on such text features, which are more explainable and more straightforward compared to the set of standard features used for this task?** Typically, computational linguists represent text with the group of its features for their classification tasks. These features could include bag-of-word representation of the text, for example. We refer to such features as "standard features" (SF) in this thesis. However, for the task of popularity prediction, linguists with little experience in the field of computational linguistics might find the manually programmed features (we name such "engineered features" as "EF") to be more intuitive, much simpler to interpret and more explainable compared to SF. For example, TF-IDF, as one of the SF of the text, is complex and built upon several mathematical formulas, which can make it hard to understand. Meanwhile, representing texts by the number of the smilies can be described and explained in a much simpler way to anyone without prior knowledge in computational linguistics.

Furthermore, if text popularity can be predicted with a high level of accuracy when using EF, these features might be used for making text predictions in other tasks. For instance, EF might be used to train a machine-learning algorithm on a database of valid and fake news to learn how to separate these categories linguistically.

- **We also aim to help other linguists to understand how to work with automatic tools for the analysis/classification of their data.** The key idea here is that our EF are created manually and, thus, can be explained in a much simpler fashion than automatically extracted features (SF). We target the problem of interpretability of textual features for classification tasks by proposing to represent texts by EF which are supposed to be more intuitive for the task at hand.

These are the steps we take to achieve our goals: (1) we first gather the textual data from Reddit forums and, (2) we train several classification models for each feature within our feature sets (we have one EF-set and three SF-sets). The set of SF features includes automatically extracted features (for example, bag-of-words or TF-IDF) which have been widely used in both text classification and programming (i.e., there are automated tools to extract data in common code libraries used for linguistics). The set of EF features contains features engineered explicitly for our task: count of words, usage of smilies, etc. Then, (3) we use two types of algorithms to build our classifiers (Naive Bayes and Support Vector Machine) and predict if comments are popular or not. Finally, (4) we show that engineered features can achieve better performance for our task, thus, supporting our idea that more straightforward features can be successfully used to represent the crucial information contained in the text. We conclude that EF can serve as a useful set of representation for text popularity prediction, based on the accuracy of the models and analysis of specific comments and their classification.

2. Background

This thesis compares two machine learning models which are described in this section. The focus of this thesis is not to go through the inner workings of the models but rather to look at various feature configurations in the respective models. That is why only a simplified explanation is given for Support Vector Machine (from now on “SVM”). The other model, Naive Bayes (“NB”) is, however, worked through in detail, because knowing how one model works can help with the intuition of how machine learning works in general.

SVM and NB were chosen because they are basic and were among the most commonly used models for text classification from the 1960s to until recently, i.e., they were standards in the field and are still used for basic tasks. Naive Bayes is one of the simplest models in the field of machine learning and SVM, like NB (as will be shown), builds its learning from prior data (Li et al., 2020).

The key concepts in the general process of machine learning are briefly described in 3.2 in order to support the explanations given in this section. A short description of Reddit is also included.

2.1. Naive Bayes

Naive Bayes is a simple probabilistic classifier that is based on the Bayes rule (Bayes 1763), which states that a particular event can be predicted given concrete conditions related to this event. One of the first papers that applied the NB algorithm to the NLP task was the 1964 paper “Inference in an Authorship Problem“ published by Mosteller and Wallace. In this paper, authors use NB to identify the authorship of The Federalist Papers (a collection of 85 articles and essays) by learning to classify each of the texts. Since then, Naive Bayes has been used to categorise text for various NLP purposes, including document classification (such as deciding if an email is a spam or not), person classification and sentiment analysis.

For the task of text classification, NB uses text representations to learn to map text with a particular label. In sentiment analysis, the words are typically transformed and considered as input features for the machine learning model. The sentiment itself would be the class label. Typically, texts are represented through the frequency of the words that they include, and the probability of assigning text to a particular label depends on the word frequency. Finally, the predicted classification is chosen based on the product of word probabilities for each label.

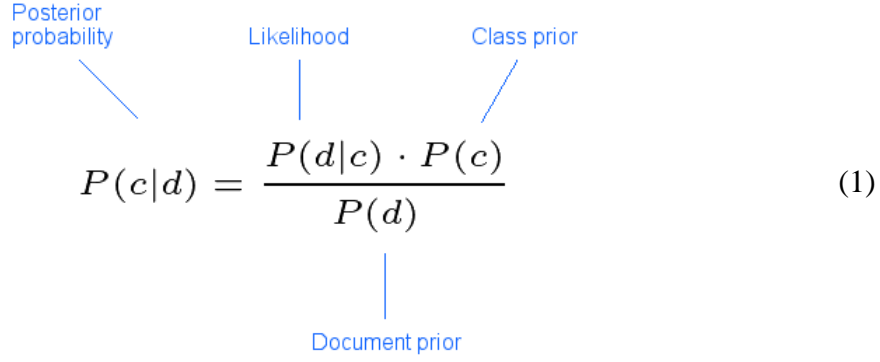
Naive Bayes is called Naive because it makes two simplifications: (1) the word order is not used as an important text feature, (2) all features are independent of each other. The first simplification means that NB does not take the fact into account that the word order could affect word semantics. For example, it is not able to learn differences in the sentiment of the two expressions: "really good" and "good, really?". The second simplification is sometimes called the *Naive Bayes assumption*. Usually, word meaning is dependent on context, i.e. the surrounding words. An example of this could be the collocation "strong tea", where "strong" cannot be replaced by "powerful" as it is not a synonym in the context of tea. NB is not able to capture such contextual dependence, which can be crucial for text classification tasks. Note that, at the same time, according to Jurafsky and Martin (2020, chapter 4), it can be computationally expensive to run a simple algorithm that would take word order and contextual knowledge into account.

NB is a straightforward and simple algorithm; however, it is typically outperformed by the logistic regression algorithm (Pranckevičius & Marcinkevičius, 2017) for text classification tasks. Since the focus of this research is to investigate the effect of using engineered features for sentiment classification rather than to identify the most effective algorithm, we will leave the investigation of the logistic classifier for our task for future work. However, note that the advantage of Naive Bayes for our study is its explainability: the inner workings of the NB algorithm are relatively easy to understand and interpret for linguists with no prior knowledge in computational linguistics. Therefore, in the following sections we are going to look closely at the Naive Bayes algorithm, analyse its mathematical foundations and provide a manual analysis of this algorithm applied to one hypothetical example from our dataset.

2.1.1. Formal definition

The core construct of the Naive Bayes algorithm is the Bayes theorem. The algorithm and its constituents will be described in detail. It will be demonstrated that the intuition behind the Naive Bayes algorithm is very simple and that it is easy to find the final probability.

The complete formula (1) is shown below. The idea behind it is to calculate the probability for a given document to be of a certain class. This is possible by breaking apart the document in pieces (e.g., its words) and collecting prior knowledge of what those pieces entail.



$$P(c|d) = \frac{P(d|c) \cdot P(c)}{P(d)} \quad (1)$$

(1) *Bayes Theorem. It states that the posterior probability of a class c , given document d , can be calculated by the likelihood of a predictor multiplied by the prior probability of the class divided by the prior probability of the predictor.*

To start with the calculations, a class must be chosen, and a new document given. Then, *class prior* indicates the proportion of that class in all prior documents. One can think of it as “contextual knowledge”, which is given in advance. This information is crucial in helping to predict the actual class of the document.

The *likelihood* asks what the probability is that the new document belongs to the chosen class. It is possible to calculate this by breaking up the new document into words and checking the prior likelihoods of those words. It would then be possible to estimate the likelihood of the whole document by looking at the words. (This will be evident in the formulas going forward and an example will be given in 2.1.6.)

The *document prior* equals the proportion of the document in all prior documents i.e., it is used to estimate the chance of a document (or its pieces) to recur. For this specific case and as suggested by Jurafsky and Martin (2020, chapter 4), this denominator will be removed from the formula since the probability of the document never changes when calculating this for different classes.

As formula (2) indicates, the prediction outcome is the class with the highest posterior probability.

$$c_{pred} = \operatorname{argmax} P(c|d) = \operatorname{argmax} \frac{P(d|c)P(c)}{P(d)} \quad (2)$$

Since the denominator appears in all posterior probabilities, scales them equally and the algorithm only focuses on predicting a class, it can be dropped. The final formula would then be the one shown in formula (3).

$$c_{pred} = \operatorname{argmax} P(d|c)P(c) \quad (3)$$

When looking at the likelihood, it is possible, as stated earlier, to break the data into several smaller pieces. Then the factors can be multiplied to get the likelihood for the whole document as seen in formula (4) and (5).

$$P(d_1, d_2, \dots, d_n|c) = P(d_1|c) \cdot P(d_2|c) \dots \cdot P(d_n|c) \quad (4)$$

or

$$P(d_1, d_2, \dots, d_n|c) = \prod_d^n P(d_n|c) \quad (5)$$

The outcome of the multiplication is the same no matter the order of the factors. The factors (d_1, d_2, d_3 , etc.) represent the words in the document, which means that the word order does not matter. Naive Bayes "thinks" of a sentence the same even though the words are scrambled, which is why Naive Bayes is said to be naive. This also means that one word does not refer or infer to another and therefore the words are independent of each other. (This will be covered in 2.1.2.)

Finally, we can combine equations (3) and (5) and produce the final formula for the algorithm, formula (6).

$$c_{pred} = \operatorname{argmax} \left(P(c) \prod_d^n P(d_n|c) \right) \quad (6)$$

2.1.2. Bag-of-words

It is possible to evaluate a document by evaluating the words contained in that document. *Bag-of-words* (referred to as “BOW” in tables and diagrams) is one of the standard and very simple text representations used for this purpose. It represents a document by the frequencies of its words. In close association with Naive Bayes, in particular formula (4), the order of the words is completely ignored, as though *each word was put and scrambled in a bag*. This allows for all sorts of classification of new documents if the words in that new document are present in prior documents.

EXAMPLE 1: SPAM CLASSIFICATION

A common text classification problem is spam classification of emails. Each email message represents a document and is either classified as Spam or Not spam. An example will be provided to show how to predict if a new email is spam or not, by breaking the email contents down to its words and looking at the prior knowledge of those words.

A spam classification is modelled by using prior knowledge of sentences known to be spam or not. The model should be able to predict if an e-mail containing a sentence is spam by applying bag-of-words to it. The sentence to be predicted is new to the model, but the words are known prior to the classification task. Here are sample sentences which, along with their label (Spam or Not spam), are known to the model prior to the classification of the new sentence:

“Don’t wait another day”	[SPAM]
“My big dream to be an astronaut”	[NOT SPAM]
“Congrats! You have won 1000”	[SPAM]

These sentences (along with their label) can be put into a table looking like this by breaking them down into their words:

#	wait	anothe	day	big	dream	astron	congrat	win	1000	Class
1	1	1	1	0	0	0	0	0	0	Spam
2	0	0	0	1	1	1	0	0	0	Not spam
3	0	0	0	0	0	0	1	1	1	Spam

Table 2.1: Bag-of-words represented in a table

In this example, not all words are fed into the table. It is common to use only the significant words in their lemmatised form in text classification. This is one way to pre-process data for the model (pre-processing will be covered in 2.3.3).

EXAMPLE 1 - CONTINUING

The new sentence is introduced to the model. The model can deduct the likelihood of that sentence to be classified as spam or not, even though the sentence is new, by looking at each word. The new sentence is:

Congrats! You have now been an astronaut for 1000 days. You win 1000!

To represent the new sentence in bag-of-words, it is broken down to its words. It is shown how by feeding the new sentence to the bottom row of the table:

#	wait	anothe	day	big	dream	astron	congrat	win	1000	Class
1	1	1	1	0	0	0	0	0	0	Spam
2	0	0	0	1	1	1	0	0	0	Not spam
3	0	0	0	0	0	0	1	1	1	Spam
4	0	0	1	0	0	1	1	1	2	?

Table 2.2: Showing new sentence at the bottom (row 4) of the bag-of-words table

Only significant words in their lemmatised form count. Each word can now be cross checked with words from the previous experiences of sentences containing the same words. In the table above, the corresponding words in the new sentence are colour coded to those in the previous sentences. It shows that four words are marked as spam (note that “1000” is present twice) and two are marked as not spam. Therefore, it is possible to make the deduction that the new sentence is spam.

This example illustrates that by looking at the words and their corresponding class, it is possible to predict a classification of an entire new sentence by the words of that sentence. This is how bag-of-words works, and it follows the Naive Bayes algorithm by formula (4) in section 2.1.1 which states that the whole is the product of its factors (here, the sentence is the whole and the words are the factors).

2.1.3. Laplace smoothing

There is a problem with introducing new words to Naive Bayes. That is because the sentence is the product of its words. A zero-encounter in prior documents means that the factor for a new document also is zero, and when multiplying anything with zero the resulting product will also be zero. That means the likelihood will be zero for the entire document. So, without earlier data to form evidence for the new document, the calculations fail, resulting in zero for all classes for the new document. This problem is often addressed as *The zero-frequency problem*. It can be avoided by implementing the *Laplace smoothing technique* (Manning et al., 2008), which incorporates a small sample correction (a pseudo count, i.e., a very small number) in all probabilities - that is, pretending that all words are seen an extra time.

EXAMPLE 2

The result could be visualized by taking Table 2 from Example 1 in section 2.1.2 and first adding an empty column filled with zeros, then adding plus one to all cells of zeros and ones, making them ones and twos. When a new word is encountered it gets associated with the new column as though it has always been there and been seen once for every sentence in the dataset. This is shown in the tables below.

If the sentence “I dream of the moon” was to be introduced, the new word “moon” is added to the table with the pseudo count of one:

#	wait	anothe	day	big	dream	astron	congrat	win	1000	moon	Class
1	1+1	1+1	1+1	0+1	0+1	0+1	0+1	0+1	0+1	0+1	Spam
2	0+1	0+1	0+1	1+1	1+1	1+1	0+1	0+1	0+1	0+1	Not spam
3	0+1	0+1	0+1	0+1	0+1	0+1	1+1	1+1	1+1	0+1	Spam

Table 2.3: Showing the addition of the Laplacian pseudo count (+1)

By adding 1 to each word above, the table looks like this:

#	wait	anothe	day	big	dream	astron	congrat	win	1000	moon	Class
1	2	2	2	1	1	1	1	1	1	1	Spam
2	1	1	1	2	2	2	1	1	1	1	Not spam
3	1	1	1	1	1	1	2	2	2	1	Spam

Table 2.4: Showing the result after summing pseudo count

Now, no words have been seen zero times, and when multiplying the word frequencies together no factor is zero; thus the zero frequency problem has been avoided.

2.1.4. Bag-of-bigrams

Although it is said that the features for the Naive Bayes classifier are independent, it is possible to try to control for a dependence between words if they are grouped together as features. Grouping words into phrases would allow us to introduce at least some level of dependence between words: words are typically used with other words to form a new meaning. Therefore, another representation that we are going to use in our modelling is *bag-of-bigrams* representation (referred to as “BOB” in tables and diagrams).

The bigram is derived from the more general term N-grams, where N is a sequence of words. 2-grams or bigrams are two-word sequences and 3-gram or trigram are three-word sequences (Jurafsky & Martin, chapter 3, 2020). For example, the sentence "I love to cook" has the following bigrams: "I love", "love to" and "to cook".

These bigrams might be used as columns instead of the words from bag-of-words to represent the sentence. By looking at Table 2 from Example 1 in 2.1.2, we get the following bigrams: "wait another", "another day", "big dream", "dream astronaut", "congrats win" and "win 1000".

EXAMPLE 3

#	wait another	another day	big dream	dream astronaut	congrats win	win 1000	Class
1	1	1	0	0	0	0	Spam
2	0	0	1	1	0	0	Not spam
3	0	0	0	0	1	1	Spam

Table 2.5: Bag-of-bigrams represented in a table

In bag-of-bigrams, this allows some context because the word "win" would not be classified as spam by itself. Only new sentences with the two consecutive words "congrats win" or "win 1000" would be classified as spam. For example, a new sentence with the words "1000 congrats" might be predicted not to be spam. As a side note, when new bigrams (such as "1000 congrats") are encountered it is possible to use the Laplace smoothing technique for the new bigrams (in the same way it would be used for bag-of-words).

2.1.5. TF-IDF

TF-IDF is the product of two calculations, TF or *Term frequency* and IDF or *Inverse document frequency*. While TF measures the simple frequency, IDF measures how much information a word provides. Combined, they define the importance (or *weight*) of a term (i.e. a word) in a document. TF-IDF takes into account that many of the words with the highest frequencies are common in every text, without being of much unique value. By taking the inverse document frequency into consideration, it avoids the problem of ranking stop words of higher importance than lexical words.

The term frequency is calculated by how many times a term is encountered divided by how many terms a document has. The inverse document frequency is the logarithmically scaled inverse of the total number of documents divided by the number of documents containing the term.

To measure frequency is so common that no single author is acknowledged for using it in or outside the domain of text classification for the first time. Spräck (1972) contributed to IDF in her paper on weighting in information retrieval, *A statistical interpretation of term specificity and its application in retrieval*.

Since TF-IDF has not been manually engineered for any feature extractions, but rather calculated by using automatic tools, the mathematics of it will not be described here.

2.1.6. An example of training Naive Bayes with bag-of-words features

To get a better understanding and insight of the mathematical formulas and how the NB algorithm works with words, an example will be provided. The main reason is to give a detailed description of how this algorithm works, along with bag-of-words, and to create an intuitive feeling of the mechanisms behind NB while simultaneously connecting it to the formulas. The example will also provide an insight as to how other features might be used to predict text popularity. For the sake of connecting the example to the goal of the thesis, online comments will be analysed. The following is an example with made-up comments, along with their popularity, from a Reddit-like forum (see 2.4.) for book recommendations. Smilies will be used as an example feature.

Example

The prior sentences are split into two sets: the training set and the testing set. (The procedure to split the prior data into a training- and a testing set will be covered in 2.3.3). The first 4 sentences in Table 2.6 below are in the training set and will be used to train the Naive Bayes model. The last, 5th, sentence belongs to the testing set. The 5th sentence will be used to test the Naive Bayes prediction capabilities in the end of this example.

	#	Sentence	Class
Train	1	It's really good 😊	Popular
	2	I want this book so bad!	Popular
	3	What a bad book	Unpopular
	4	They say this book is good, really?!	Unpopular
Test	5	Good! I want it! 😊	?

Table 2.6: Example table of forum comments split into training- and testing sets along with their class.

First, note that it is possible to get an intuitive feeling about the popularity label by looking at the sentences. For example, the first training example (row #1) has a positive smiley in it, which makes it likely that this example is popular. Also, the last sentence in the training set ends with “?!” punctuation marks, which might be helpful as well. However, in this example only smilies will be used in determining the popularity of the text, as they are shown to have a sentimental value in text classification by Kralj Novak et al. (2015:12).

Consider the test sentence “Good! I want it! 😊” (row #5). What could the label be for this sentence? By looking back at the training set, row #1 tells us that the word “good”, as well as the symbol “😊”, is labelled popular and row #2 tells us that “want” is labelled popular, although “good” is also found on row #3 which has an unpopular label. Overall, most words point to the fact that the sentence is popular which is why the whole sentence is popular (this is how Naive Bayes “reasons”).

It will now be demonstrated that this is indeed the case for Naive Bayes if the sentence on row #5 was to be plugged into the formula (1) from the formal description in 2.1.1. This will result in two formulas because both classes "popular" and "unpopular", i.e., both labels, need to be considered.

To calculate the probability of the posterior class for the sentence at row #5, the class priors and the likelihoods are needed.

The class priors

$$P(pop) = \frac{2}{4} = \frac{1}{2}$$

$$P(unp) = \frac{2}{4} = \frac{1}{2}$$

The likelihoods for the words

$$P(good|pop) = \frac{1}{10} + \frac{10}{10} = \frac{11}{10}$$

$$P(good|unp) = \frac{1}{10} + \frac{11}{11} = \frac{12}{11}$$

$$P(i|pop) = \frac{1}{10} + \frac{10}{10} = \frac{11}{10}$$

$$P(i|unp) = \frac{0}{10} + \frac{11}{11} = \frac{11}{11}$$

$$P(want|pop) = \frac{1}{10} + \frac{10}{10} = \frac{11}{10}$$

$$P(want|unp) = \frac{0}{10} + \frac{11}{11} = \frac{11}{11}$$

$$P(it|pop) = \frac{1}{10} + \frac{10}{10} = \frac{11}{10}$$

$$P(it|unp) = \frac{0}{10} + \frac{11}{11} = \frac{11}{11}$$

$$P(\bigcirc|pop) = \frac{1}{10} + \frac{10}{10} = \frac{11}{10}$$

$$P(\bigcirc|unp) = \frac{0}{10} + \frac{11}{11} = \frac{11}{11}$$

The likelihoods are calculated by looking at how many times each word appear in the different classes divided by the number of different words in that class. The popular class has 10 words, and the unpopular class has 11 words.

Note that they are also added with the Laplacian smoothing value of 1 (or 10/10 in the popular and 11/11 in the unpopular, which is the same).

By formula (4) in 2.1.1, the likelihood for the whole sentence can be composed by multiplying the likelihood for the words.

Likelihoods for the whole sentence:

$$P(\text{goodiwantit}|\text{pop}) = \frac{11}{10} \cdot \frac{11}{10} \cdot \frac{11}{10} \cdot \frac{11}{10} \cdot \frac{11}{10} \cdot = \frac{161051}{10000} \approx 1.61$$

$$P(\text{goodiwantit}|\text{unp}) = \frac{12}{11} \cdot \frac{11}{11} \cdot \frac{11}{11} \cdot \frac{11}{11} \cdot \frac{11}{11} \cdot = \frac{175692}{10000} \approx 1.09$$

The probability of each posterior class can now be calculated.

$$P(\text{pop}|\text{goodiwantit}) = P(\text{pop}) \cdot P(\text{goodiwantit}|\text{pop}) = \frac{1}{2} \cdot 1.61 = 0.805$$

$$P(\text{unp}|\text{goodiwantit}) = P(\text{unp}) \cdot P(\text{goodiwantit}|\text{unp}) = \frac{1}{2} \cdot 1.09 = 0.545$$

By formula (6), the class associated with the highest posterior class of all classes is the predicted class by Naive Bayes. 0.805 is larger than 0.545, therefore the sentence "Good! I want it! 😊" is predicted to be popular.

2.2. Support Vector Machine (SVM)

The support vector machine is a machine learning algorithm developed by AT&T Bell laboratories by Vapnik and his colleagues (Boser et al., 1992). For example, it has been used successfully in text classification to show that the real world can be reflected by Twitter texts (Aramaki et al., 2011). SVM predicts the classes for the sentences in Table 2.6 by trying to divide the dots with a line. The colours of the dots in Figure 2.2 represent the two labels. The points are defined by their features in a multidimensional space. If a point has two features, it can be placed in a 2-dimensional space by the measurements in those two features. For example, for the two features “Similarity” (i.e. meaning how similar the sentence is with the other sentences) and “Special words” (i.e. how many technical terms the sentence has) the sentences can be plotted in a 2-dimensional graph like below.

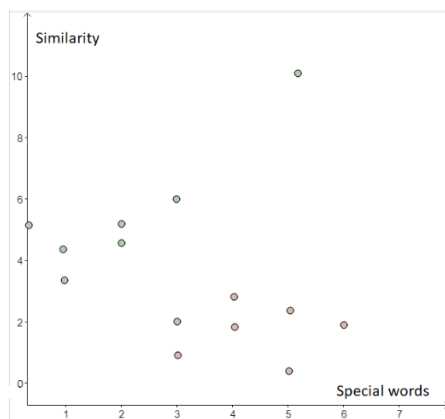


Fig 2.1: The sentences can be represented by dots in the 2-dimensional graph

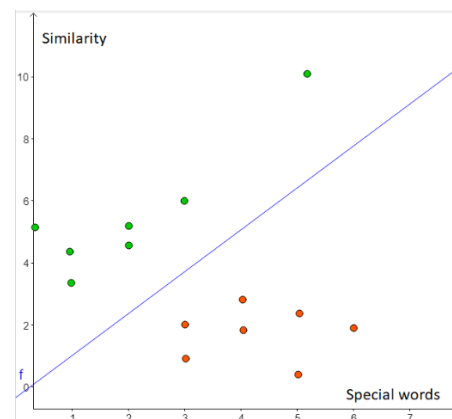


Fig 2.2: The dots are divided by a line

SVM predicts the classes for the sentences above by trying to divide the dots with a line. The dots representing the sentences on one side is predicted to be of one class and the dots on the other side another class. The colours of the dots in Figure 2.2 shows the two classifications in a binary classification.

SVM can make predictions about any number of features. If three features were used a 3-dimensional space would be required and the divisor for the predictions would be a plane instead of a line. In general, N features require a $(N-1)$ -dimensional divisor.

2.3. Process of machine learning

The process of machine learning will be described in three parts. It is a general description only, why no references will be given.

2.3.1. Getting data

A lot of data is usually needed to train a model with machine learning. The amount of data needed depends on what the model should learn to predict and which algorithm the model is based on though, as a rule, more data equals better prediction. After data retrieval, the process is followed by statistical analysis of the dataset. This is done to better understand the target of prediction by finding out which features stand out, which features to measure as well as which model to choose.

2.3.2. Feature extraction

Most feature extraction tools are readily available in programming packages, e.g. as those provided by Sci-kit learn (Pedregosa et al., 2011). Some feature extraction algorithms might need to be developed and then tested. This can be done by first (1) generating a smaller test sample (which should be designed to yield a specific score for a specific algorithm) and then (2) feeding the samples onto the algorithms and scoring them. A table to keep scores might look like this:

	Feature 1	Feature 2	Feature 3
Sample #1	2	0	0
Sample #2	0	3	0
Sample #3	0	0	5
Sample #4	1	2	1

Table 2.7: Feature extraction algorithm test table

The numbers in the table indicate the sample's test score and the feature extraction algorithm. The feature extraction algorithm is done when (3) the sample yields the intended score for that specific feature. Better test results at this stage provide better feature extractions, which in turn affects the whole machine learning prediction capability.

2.3.3. Training and validating the model

With data, feature extraction and a chosen model, the data gets scored feature by feature and is then split into two sets; a *training set* and a *testing set*. The training set includes the correct answers and is used to train the model (i.e., the model is *fitted*, which means that the weights

are set so that the measured test data results in the correct answers). The testing set is then fed to the model to see what it predicts. The model does not know the correct answer to the testing set; thus, it can be validated. The *accuracy* is calculated by the ratio of the correct answers and the number of answers in total.

2.3.4. Some definitions

Label

The label is the class or score which is assigned to a sample in the dataset. In popularity prediction classification, the labels can be "popular" and "unpopular", meaning that each sample in the dataset is either popular or unpopular.

Binning

Binning means to put data into several bins corresponding to their label. When labels span over a continuous range, binning is needed to bind data to their respective classes. Two bins must be constructed for binary classification, three bins for ternary, and so forth.

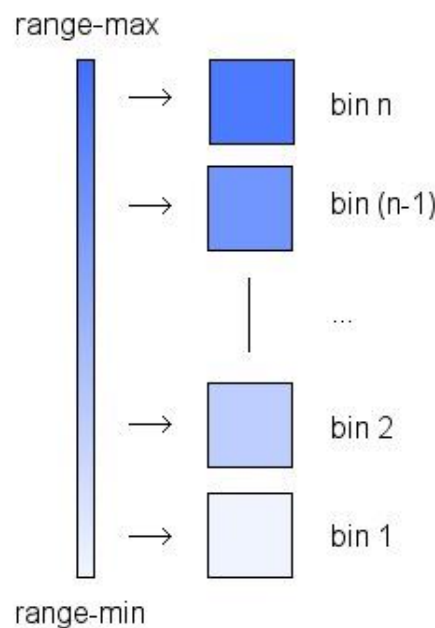


Fig 2.3: The picture shows how to put an arbitrary range of labels into discretised bins

Scaling

Some feature scores might have a larger range than others. For example, one feature might count the number of characters in a sentence, and another one all proper nouns. The count of characters might get larger by one magnitude compared to the count of the nouns. Some models cannot handle features of different magnitudes well – the model amplifies the meaning of the larger feature scores. A solution is to scale all features down to the same range (or in close vicinity of the same range).

Character count	Proper noun count		Character count	Proper noun count
26	2	→	0.26	0.2
156	5	→	1.56	0.5

Table 2.8: Scaling

K-Folds and cross validation

When training and validating a model, it can sometimes yield better accuracy if the model can train and be validated several times on different testing sets. By this process, it is said that the data is *folded K* times (meaning the dataset is shuffled for new sets K times) to allow for *cross validation* between the K number of testing sets.

2.4. Reddit

Because this thesis evolves around Reddit comments and Reddit uses a special rule set for comments which in return render them popular or not, a brief explanation will be provided for Reddit, its parts, and how it functions.

Reddit is an online discussion board (or forum) with many groups of topics called *subreddits*. All subreddits are entirely independent and usually have a theme, such as politics or cats. Participants on the subreddit create various topics on the theme in which discussions take place. In a politics subreddit, a topic could be "The 2020 US election". Participants discuss with each other by creating *comments*.

Participants can express a positive or negative sentiment about other participants comments, which is done by clicking an upwards turning arrow (positive) or a downwards turning arrow (negative). The positive and negative votes equal a popularity rating, which in this thesis is called *score*.

3. Methodology

It was decided to create a customised scraping tool to build a corpus of extracted Reddit comments. The tool is described in section 3.1.2. The comments were scraped from a randomly chosen set of subreddits and data has been merged from two points in time. The dataset was analysed and experiments (section 3.2.) were made to test if various assumptions would hold true and to overcome certain problems. The experience from this resulted in a set of EF which will be covered in section 3.3.

The two models SVM and NB have been trained and validated with the data that was scraped. EF and SF have been selected for use in both cases. This results in eight cases of model groups in which a further division of three was made to ultimately reach 24 trained models.

A few models emerged as more interesting than others, why the thesis focuses on these. For these models, confusion matrices have been calculated and sample sentences have been drawn from the testing set for each matrix in order to prove statements about the different models and their results (i.e., their accuracies).

3.1. The dataset

Two datasets have been scraped off Reddit. The first dataset of 43 562 comments was scraped 5th October 2020 and the second of 41 711 comments 11th November 2020. These datasets have been merged into a single, larger dataset. There are two main reasons for this merge:

- 1) To remove bias/typicality regarding the time the comments were scraped (for example, the US Presidential Election spawned a lot of discussions on that topic on Reddit at the end of November 2020).
- 2) To fetch more outliers such as comments with particularly low or high score. Doing this helps the training.

The smaller trimmed sample of this merged dataset, called *the trimmed set*, has been used for training the model. The same number of comments from both original datasets were trimmed without taking away any outliers (to satisfy the goals 1 and 2 for creating the merged dataset).

After merging the datasets, the merged dataset was trimmed in order to make it more computable. The merged dataset contained over 250 000 unique tokens which creates feature tables with more than 2 billion cells. With an average word length of five characters this amounts to 10GB of data to work with and would require the double working memory from the computer. The motivation to lower the required working memory is to show that good results can be achieved using computers available to most linguists, such as laptops.

The trimmed dataset contains 16 840 comments from the subreddits displayed in the table below. The subreddits were randomly chosen from the 2020 list of the 31 subreddits with the highest number of subscribers (Baer, D., 2020).

Subreddit	Number of comments
askreddit	9427
askscience	613
explainlikeimfive	587
news	2269
science	896
todayilearned	1389
worldnews	1659
TOTAL	16840

Table 3.1: Comment distribution among subreddits in the trimmed dataset.

3.1.1. Statistics

This section shows graphs about the dataset, e.g., the distribution of the comments scores and characteristics such as how many significant words, stop words or positive smilies exist for all comments. The features measured here are strongly associated with the EF, which are described in detail in 3.3. By viewing these graphs it is possible to (before the machine learning) get a sense of which features correlate with high and low scores.

Number of comments	16 840
Number of words	860 211
Mean words per comment	~ 51

Table 3.2: number of comments and words and mean words per comment

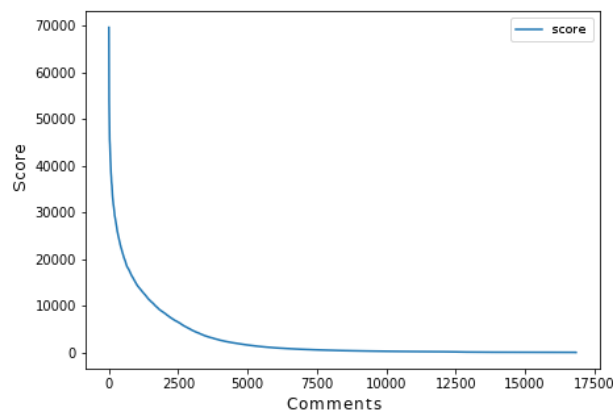


Fig 3.1: Distribution of scores

The distribution shows that very few comments have high score, and almost half of the comments score around zero. Only ~2500 comments have a score above 10 000.

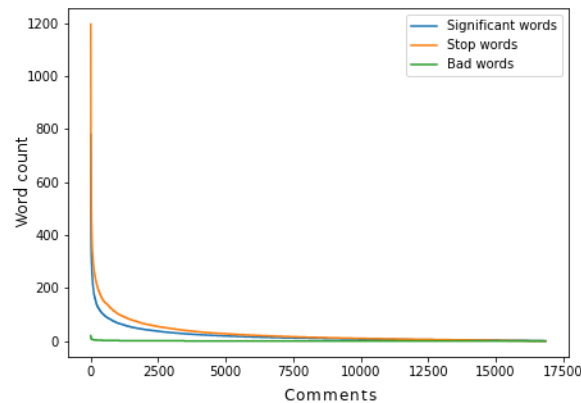


Fig 3.2: Distribution of word counts in each comment

A very small number of comments have a significant number of bad words (noticeable at the start of the green line) and in general few comments are long. The typical length of a comment is close to zero words.

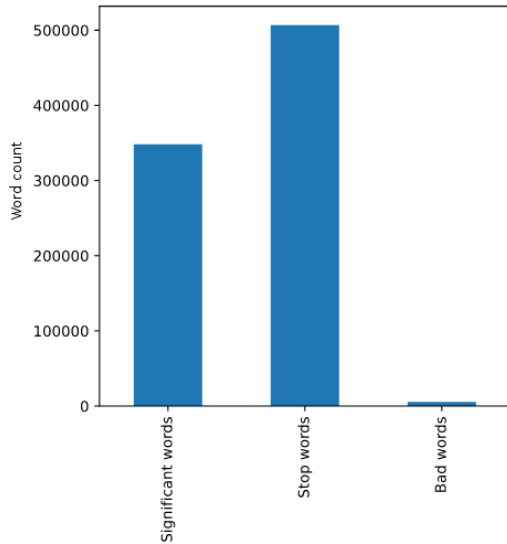


Fig 3.3: Word count bar chart

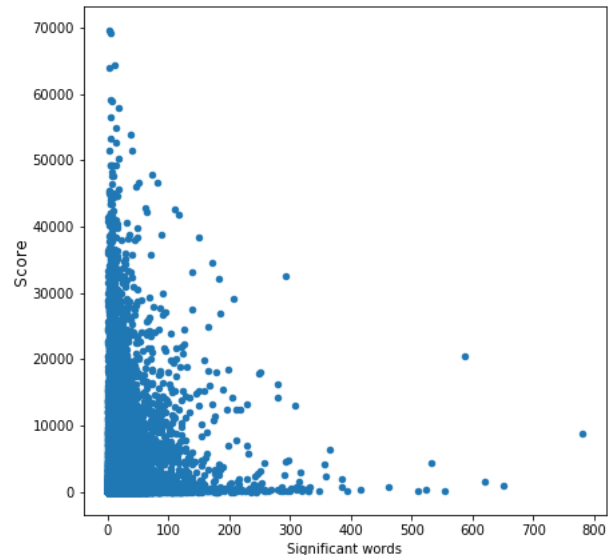


Fig 3.4: Significant words scatter plot

There are more stop words in total than significant words. Bad words are almost non-existent.

The number of significant words in a comment are spread in comments with all scores. The trend shows that the more significant words exist in a comment, the lower the comments score is, but there are a few outliers.

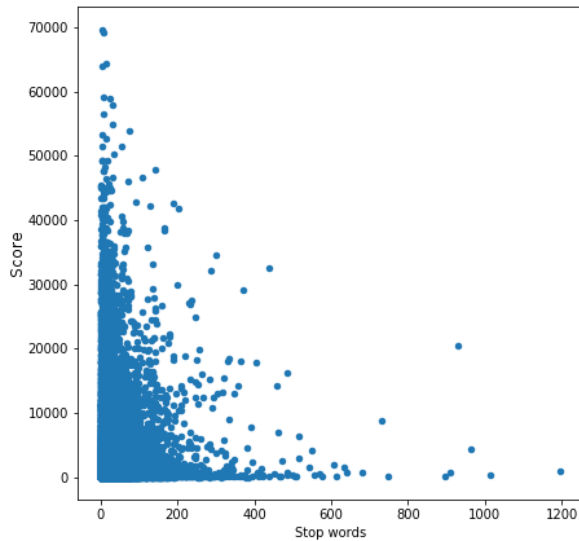


Fig 3.5: Stop words scatter plot

Stop words follow the same pattern as significant words. However, it should be noted that some comments have a lot more stop words than significant words and still seem to receive a remarkable score (For example, the top rightmost dot almost has 1000 stop words – comparing this to the significant words scatter graph, it correlates to a comment with a maximum of 600 significant words).

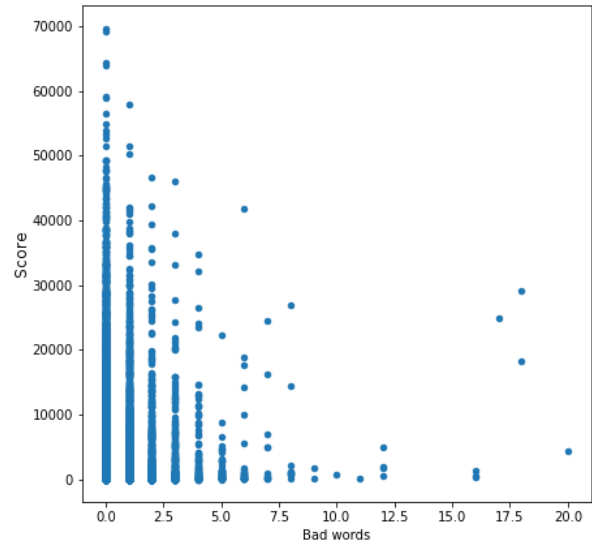


Fig 3.6: Bad words scatter plot

It seems possible for comments to have bad words in them in all score ranges, but the trend seems to show that the more bad words a comment has, the lower the score is for that comment.

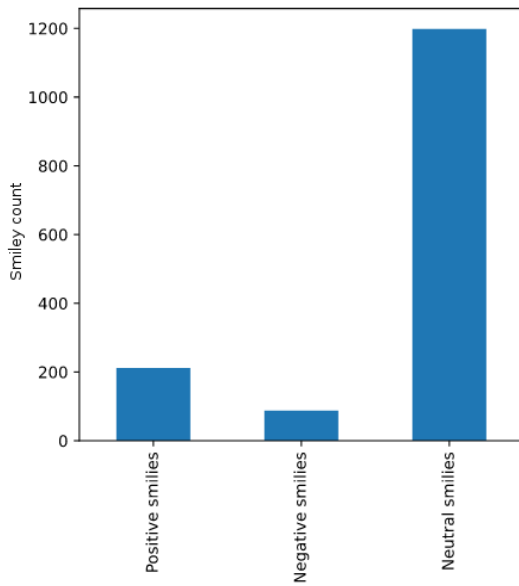


Fig 3.7: Smiley count bar chart

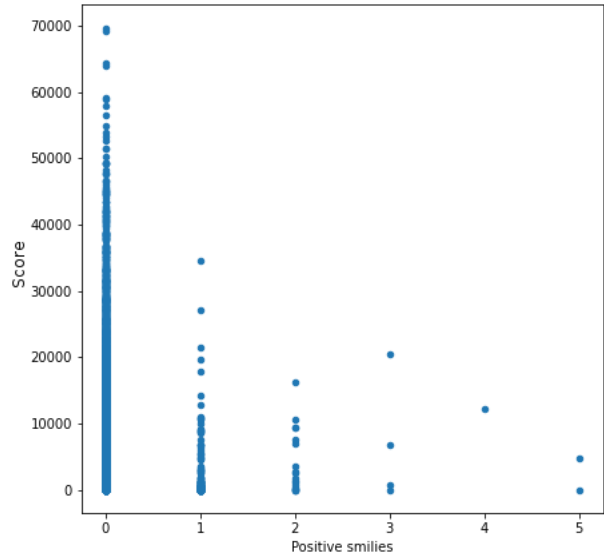


Fig 3.8: Positive smilies scatter plot

Smilies are not uncommon, but the number of each type are highly different. In the 16840 comments there are only about 200 positive smilies compared to 100 negative and as many as 1200 neutral smilies.

Most comments have zero positive smilies and it doesn't seem to affect the score much. The comments with positive smilies are almost countable. Only seven comments have more than two positive smilies.

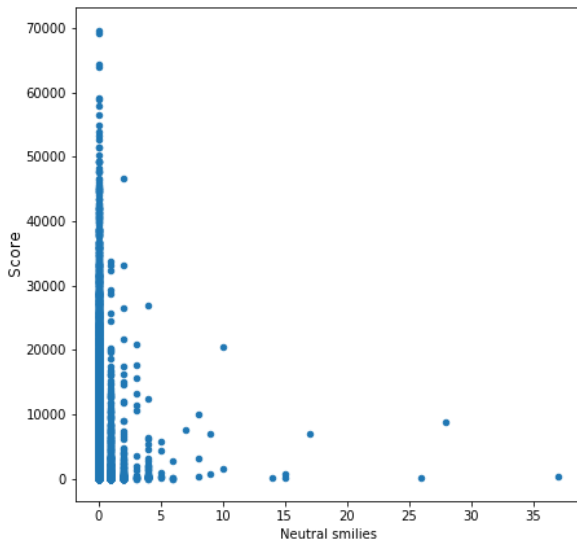


Fig 3.9: Neutral smilies scatter plot

The trend in the graph shows that comments with more neutral smilies yield lower score. Most of the comments with above 10 neutral smilies have a score of around zero.

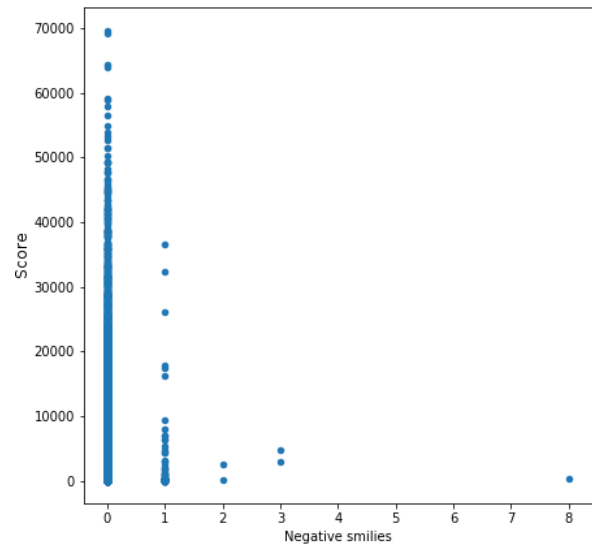


Fig 3.10: Negative smilies scatter plot

Negative smilies are rare. With more than one negative smiley no comment reaches close to the mid-tier score comments. One outlier comment with close to zero score has as many as eight negative smilies.

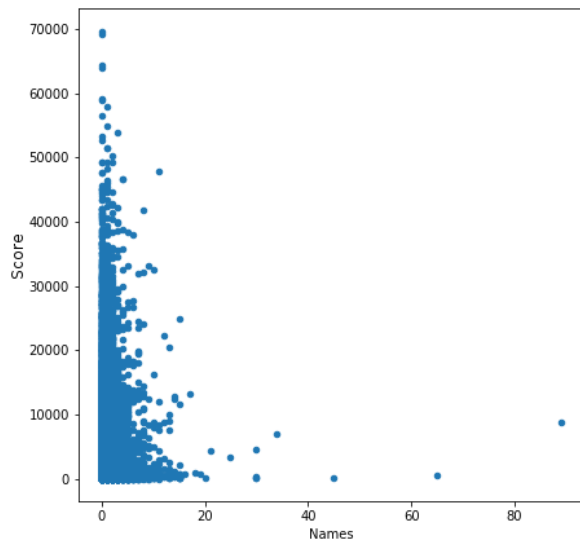


Fig 3.11: Names scatter plot

Person names are popular in comments. A significant number of comments include a name and some as many as 20 names. One outlier comment has around 90 names in its comment with a somewhat remarkable score of 10 000.

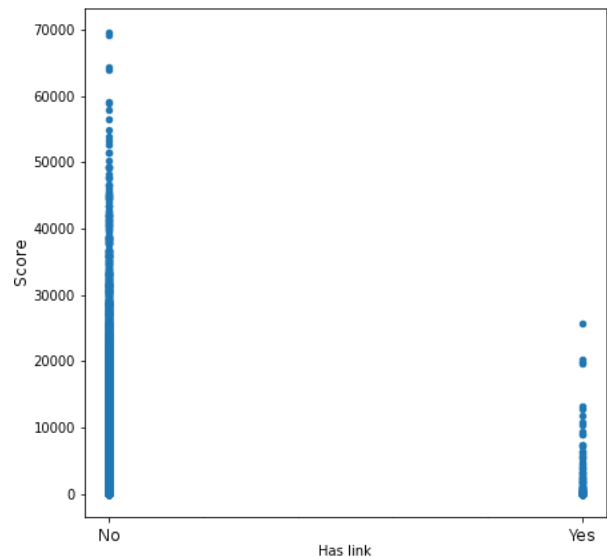


Fig 3.12: Link scatter plot

Most comments do not contain a link. Typically, high score comments do not contain a link, however a larger set of comments that do contain a link have a significant score above the close-zero-range.

3.1.2. About the scraping tool

Although many large datasets for Reddit comments exist and fit the research subject; all of them are either very large or limited to earlier time frames or certain subreddits. Also, they include noise (manipulated data) or metadata.

For this thesis, the relevant scope for each comments data resides in its text body, its linguistic features and *some* context (meaning the comments parent topic and the sibling comments in the same topic when checking similarity scores, see 3.1.1.).

This research does not focus on metadata (such as author membership, timestamps, earlier author awards and achievements, or subreddit rules, etc.) because these are not linguistic features. Therefore a customised scraping tool (*parseit*, Leppänen 2020) has been programmed to fetch only relevant data from various subreddits and to build a corpus.

The tool focuses only on the body and its label and uses the known library PRAW (Boe et al., 2016) and python Reddit API wrapper. The scraping tool works by selecting a list of subreddits and a number of topics. It then fetches that number of comments for each topic and stores them into a data file.

3.2. Experiments

This section will cover the process of feature extraction and machine-learning for this thesis, as well as some problems that have required specialised measures or workarounds. The overall process has followed the key points described in 2.3.

Scraping of data is described in brief in 3.1.2. One of the better alternatives for filtering of Reddit comments is to sort by monthly high score topics (which in general has more comments than low score topics do). Many filtering methods for the scraping tool was tried but did not result in as many comments, although the comments had better variety in their score range. To get more comments and with better variety of scores, scraping by monthly filtering was made twice with a month apart from each other and then a multitude of comments with the same score (close the zero-range) was cut out (see 3.2.1.).

To follow the goal of this thesis, as many automated and easy tools as possible are used. For example, all feature extraction algorithms uses the CountVectorizer except the TF-IDF features which uses TfidfVectorizer. Both are functions in the standard Sci-kit tool-set. To calculate the similarity metrics, TF-IDF has been used along with Scikit-learn's common pairwise metric, the cosine similarity metric (Scikit-learn, 2011).

MultinomialNB and SVMLinear model implementations were specifically chosen for NB and SVM respectively, because Sci-kit offers them in their standard library and are the easiest ones to work with.

For binning, two, three and four bins were chosen to train models on to allow the models a little flexibility on labeling the comments. The motivation for more than just a binary classification is given in the discussion.

3.2.1. Dataset cut out

The voting system for Reddit comments makes it prone to virality. A comment which gains a high enough score gets more traction because Reddit’s internal algorithms favour it. For this reason, outlier comments are vulnerable because votes for an already upvoted comment often seem to accelerate. This resulted in an unbalanced dataset. Very few comments ($n < 3000-5000$) had any score not close to the zero-range, see Fig 3.13.

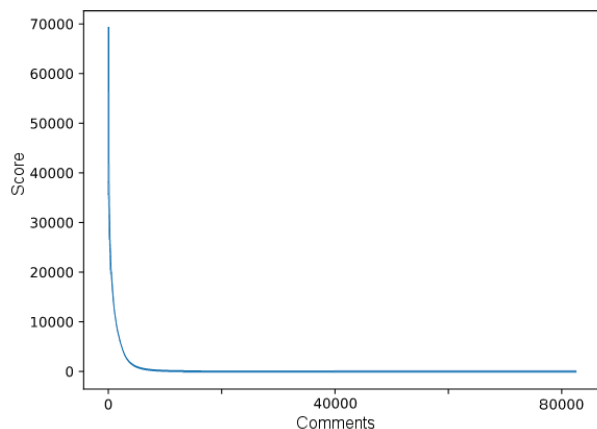


Fig 3.13: The score distribution of all scraped comments

This problem was solved by two means primarily: (1) removing 80% of the comments close to the zero-range (resulting in the *trimmed set*, see 3.1.) and (2) choosing a binning strategy called “kmeans” (as explained by Demonstrating the different strategies of KBinsDiscretizer, 2020) which handles unbalanced scores better. However, these two steps proved not to be enough. After the binning, the comment scores were still too unbalanced, as seen in figure 3.14, which shows that the lower shelves (the bins) are longer than the ones above.

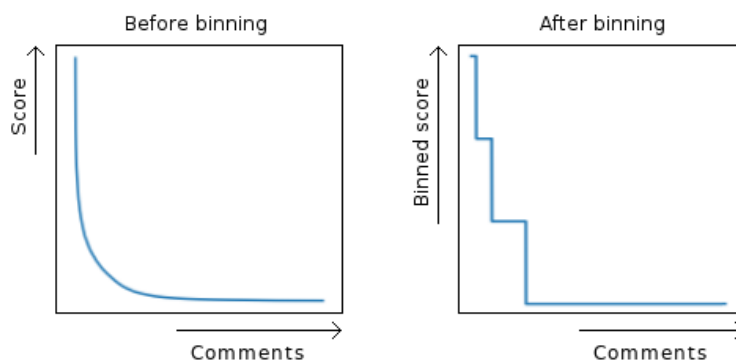


Figure 3.14. Before and after binning

A third measure was needed before training the models: a cut off from the bins. An algorithm to cut the bins was created and worked by the following:

1. Calculate maximum bin size for even distribution. For four bins this is $100/4 = 25\%$
2. Drop off the comments with the lowest scores for each bin until the size is 25%. The bins that have a smaller size than 25% are not affected.

The number of times this algorithm was run on the binned dataset was experimented with and resulted in four times for most models. The process can be visualised as per figure 3.15. This cut out was enough to train the models sufficiently.

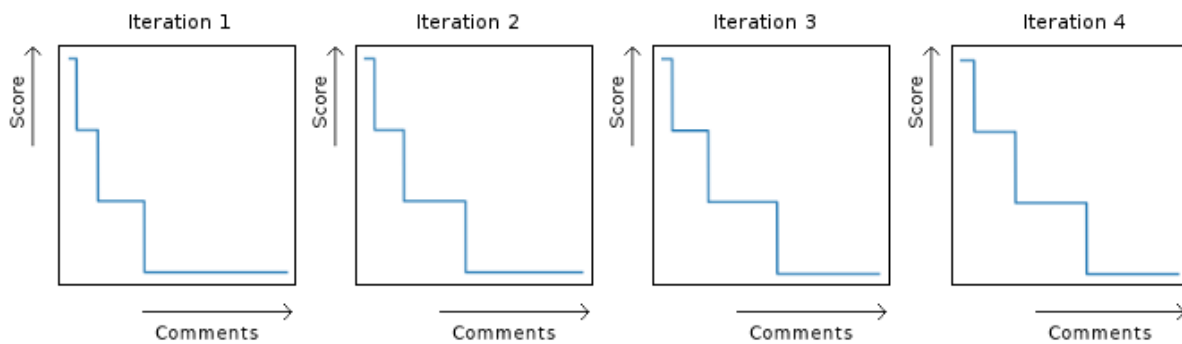


Figure 3.15: Bin-cutting algorithm

3.3. Features

3.3.1. Definitions

Score

A user-generated comments' number of votes, ranging from downvotes (negative score) to upvotes (positive score). If the score is zero, the comment has not received any votes.

Token

A string of characters separated from surrounding characters by spaces. A token is often described as a regular word but can also be a compound between a word and characters, such as "No!" (here, the exclamation mark is part of the token). A token can also consist of special characters without any recognisable word, such as a smiley, e.g., "😊" or ":").

(This is not a strict definition, but it is the definition that will be used in this paper. The definition varies.)

Stripping

When a token is stripped from one or more characters it means that those characters are removed from the token.

Ignored characters set

Some characters are stripped from a token so that the token can be more easily lemmatised. This set contains the characters which are stripped from the tokens. Examples include the punctuations and characters often denoted as "special characters", i.e. those found in the string `#$%!-~`. Characters which are used for formatting in digital documents, such as `\n` and `\r`, are also included. Examples:

`cost? → cost`

`blue. → blue`

`books, → books`

Lemmatisation

This is the process of generating a lemma from a token. The token is first stripped of all characters in the ignored characters set. Then the token is PoS-tagged (the Part of Speech tag is associated with the token) if it is possible. The token is then transformed into its lemmatised form, i.e. the lemma. This process uses WordnetLemmatizer (Bird et al., 2009) with its English vocabulary and PoS-functionality. Examples:

is? → be

are → be

t-shirts! → t-shirt

Vocabulary set

The vocabulary set is the set which contains all tokens that are encountered in the dataset, after the tokens are stripped from the set of tokens to be ignored and thereafter lemmatised.

Stop words set

The stop words are a set of words which contains function words and other words that are usual to ignore in language processing. These are usually pronouns, prepositions, help verbs, numbers (ordinal/cardinal), some adverbs and some abbreviations/acronyms. The exact list is taken from Scikit-learn (Pedregosa et al., 2011) datafiles, which gets their lists from Stop Word Lists in Free Open-source Software Packages (Nothman et al., 2018).

Bad words set

The bad words set contains words and phrases that are inappropriate in most social settings, such as profane words. This kind of list is sometimes used for automatically censoring forums or other social media platforms, as well as for email filtering (i.e., in spam filters). Examples of words include "barf", "god", and "vulgar" (it is up to the imagination of the reader to come up with other examples). Purposely misspelt words are also included, such as sh!t. The exact list is taken from Bad Words List and Page Moderation Words List for Facebook (Free web headers, n.d.)

The smilies set

The smilies set comes from the paper Sentiment of Emojis (Kralj Novak et al., 2015) and is divided into three smaller sets (positive, neutral, and negative) based on the findings of that paper. The paper only analysed Unicode smilies (symbol characters outside the ASCII character set), so a few common two- or three-character smilies were added to these sets. For example, :D was added to the positive set, :/ was added to the neutral set, and :(was added to the negative set.

3.3.2. SF-sets

In this research, we use bag-of-words, bag-of-bigrams and TF-IDF (see chapter 2) as feature sets. All three SF-sets use the programming library Sci-kit learn (Pedregosa et al. , 2011) to construct their features, which uses words, bigrams and the TF-IDF score for each set, respectively. The set of features are limited to 6000 (compared to the 11 in the EF-set) to lower

the requirements of the machine which computes them. To remove words which are not significant, stop words from the *stop words set* are used.

3.3.3. The EF-set

The EF-set contains 11 features described in detail below. They are engineered and tested in the way described in 2.3.2. At the end of this section a table with all the features will be presented along with some of the sentences used to test the feature extraction algorithms.

Significant words count

This feature counts the sum of significant words, i.e., lexical words; tokens from the vocabulary set not encountered in stop words or bad words set. Examples include "jump", "red" and "door". The purpose of using word count as a feature is to see if and how a comments word quantity affects its voting score.

Bad words count

This feature counts the sum of the tokens from the bad words set. The motivation for using bad words count as a feature is that profane words might affect comment score. In the dataset, it is also possible to see a correlation between bad words and comment score (label).

Stop words count

This feature counts the sum of the tokens from the stop words set. The motivation for using stop word count as a feature is that too many stop words might render a comment uninteresting, therefore not generating as high score as comments with fewer stop words.

Smilies count (three features)

These are three separate features which all count the smilies from the smilies set. Distinguished from one another (as they are also engineered), they count the positive smilies, the neutral smilies, and the negative smilies, respectively. The motivation for using smilies is that smilies in general help to reflect the author's standpoint/view and might make the author's sentiment clearer (Kralj Novak et al., 2015:12).

Person name count

This feature counts the names from the names set. Some subreddit communities encourage comments backed by references why using names in comments might (depending on the context) increase one's validity. Person names can also stand out as a visual element in a text because of the first capital letter. Because referencing names is sometimes encouraged, this might yield positive or negative voting from a user, which is the motivation for this feature.

Has link

This feature checks if a comment contains a link; in particular the string “http” which covers links starting `http://` as well as `https://`. Including links in comments might reflect upon the score, as some subreddit communities encourage that comments contain references. Furthermore, a comment with an embedded hyperlink often stands out because links are usually colour coded and therefore might attract focus, which might yield votes from users.

Similarity with topic

This feature checks the comments similarity with the topic by checking the cosine similarity between the comment’s TF-IDF and the topics’ (see description of topic in 2.4.1) TF-IDF for all lexical words. By good *netiquette* (Internet etiquette) it is usually considered rude not to follow the original posters topic. The TF-IDF similarity checks for similar words, which is why this feature is relevant.

Example: A new commenter hijacks the original posters talking point by creating a new comment which does not have anything in common with the topic. The new content is said to be *off-topic* and is considered rude because the hijacker uses the original posters high score as a gateway to more readers.

Similarity with all other comments

This feature examines the comments similarity with all other comments. In particular, it checks the vocabulary set (which is built from all the tokens in the comments) by checking similarity between the comment and the vocabulary set’s TF-IDF vectors. The motivation is the same as for the Similarity with topic feature; however, it allows the comment to sway further away from the topic (and still be regarded as *on topic*) as long as the words used in the content are similar to what other commenters use. The value is the mean value of the comment and each other comment in the topic (i.e., for three comments, the first comment would have the mean value of the similarities checked with both the second and third comment).

TF-IDF mean value

This feature is almost the same as the feature Similarity with all other comments. However, it does not check for similarity or use the vocabulary set, which gives a slightly different result. The motivation for this feature is that it is much easier to write and might be better scaled from the start which might help when training the model. The value is calculated from the mean value of all words TF-IDF values.

Example

The following example shows all EF put together in a table as a feature set. The comments in the first column present a sample set of the comments that the features were tested against when engineering them (see 2.3.2). The highlighted cells show which features the comment was created to test.

Topic	The authors son has released a new fantasy series! The first book is about the Blue wizards! OMG YOU HAVE TO BUY :D:D
--------------	---

The topic is the same for all comments.

#	Comment body	Similarity with topic	Similarity with other comments	TF-IDF mean value	Significant words count	Stop words count	Bad words count	Positive smiles count	Negative smiles count	Neutral smiles count	Name Count	Has link
1	Fantasy is boring :/. I'd rather dive into some Nietzsche and I think everyone should	0.09	0.21	0.16	6	9	0	0	0	1	1	0
2	No way! The blue wizards finally got a book?? OMG I have to buy it xD 🤖	0.15	0.30	0.19	7	8	1	1	0	1	0	0
4	I hate this author :C he is really bad he can go and ugly himself ugly	0.11	0.21	0.13	4	10	2	0	1	0	0	0
5	this is an old bag with barf book 🤢	0.12	0.18	0.20	3	5	2	0	1	0	0	0
6	if you are going to buy this book, make sure you get it at towns hall because they have extra t-shirts!	0.09	0.26	0.12	6	15	0	0	0	0	0	0
7	how much does it cost?	0.00	0.11	0.20	1	4	0	0	0	0	0	0
8	BLUE. JUNO. 100% sh!t http://example.com	0.00	0.22	0.45	4	0	1	0	0	1	0	1
9	I can pay \$\$\$	0.00	0.16	0.35	2	2	0	0	0	0	0	0
10	The authors son has released a new fantasy series! The first book is about the Blue wizards! OMG YOU HAVE TO BUY :D:D	1.00	0.32	0.14	10	12	1	2	0	0	1	0
11	Hey Tommy the blue, this is a great book for you 😊❤️	0.08	0.25	0.20	6	6	0	2	0	0	1	0
12	marry fantasy blue Nietzsche awesome Tommy trolololo	0.18	0.26	0.38	7	0	0	0	0	0	2	0
13	author, blue wizard	0.12	0.17	0.57	3	0	0	0	0	0	0	0
14	Tommy, blue, awesome book, ugly, fantasy, amazing, author	0.08	0.30	0.40	7	0	0	0	0	0	0	0

Table 3.1: EF combined along with their scores from test sentences

Sentence #6 was written specifically to test the feature extraction algorithms *Significant words count* and *Stop words count*. There are 6 significant words and 15 stop words in this sentence. Other noteworthy sentences are #10 and #14, which show high score for *Similarity with topic* and *Similarity with all other comments*. Sentence #7 works as an inverse test, meaning that it not should result in any significant score for any column (i.e., it has very low scores for each feature).

4. Results

In total, 24 models have been trained: one for each classifier, each set of features, and for the binning of two, three and four bins. The results are presented in Table 4.1.

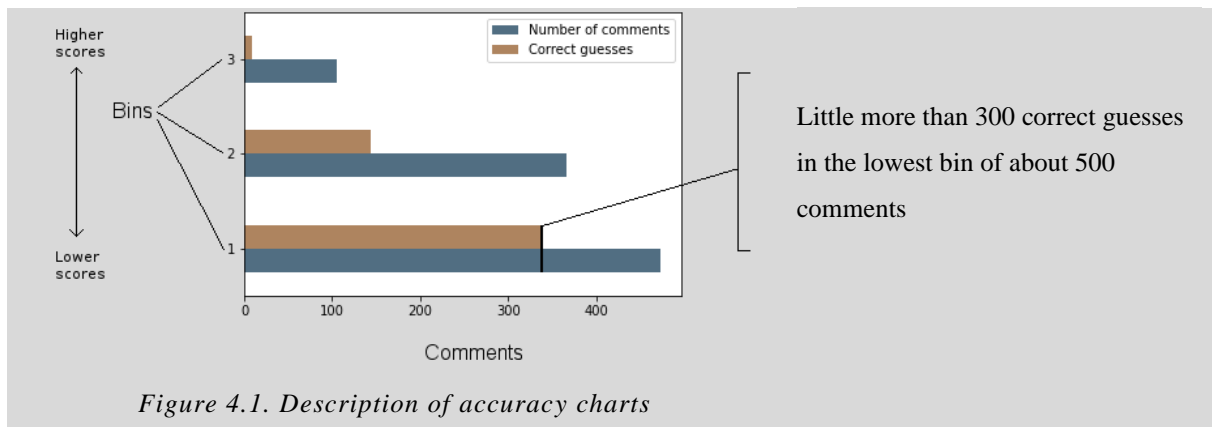
Feature set	Naive Baves			Support Vector Machine		
	2 bins	3 bins	4 bins	2 bins	3 bins	4 bins
Bag-of-words	65.49	51.74	43.40	62.31	48.97	40.37
Bag-of-bigrams	65.88	49.73	41.71	64.49	48.59	40.37
TF-IDF	67.26	52.29	44.23	63.83	50.60	41.31
Engineered features	67.01	58.07	47.44	86.65	74.32	61.64

Table 4.1: The accuracies of trained models. All numbers represent %-accuracy of their prediction.

The best accuracy for each model and bin is written in bold.

4.1. Accuracy bar charts

On the following pages, horizontal bar charts are shown for each trained model. In the graph there is a group of bars for each bin. The lower bar in the group shows the number of comments in that bin and the higher bar shows the correct number of guesses. The first bin contains the comments with the lowest scores and the last bin contains the comments with the highest scores. The bar charts are described in Figure 4.1.



The reason why the graphs are important is to show whether a model learned to predict comments in different bins and not only in the lowest bin. For example, if there were 70 comments in the lowest bin and 30 comments in the highest bin, a model which predicts that all comments are in the lowest bin would score as high as 70%. Therefore, the accuracy from Table 5 alone does not fully explain the models' prediction ability.

Naive Bayes - Bag-of-words

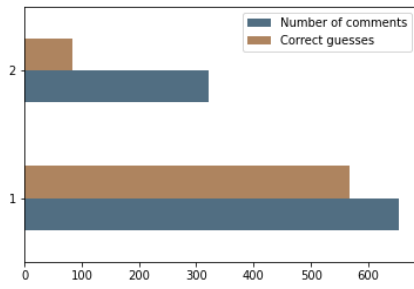


Fig 4.2: NBBOW2

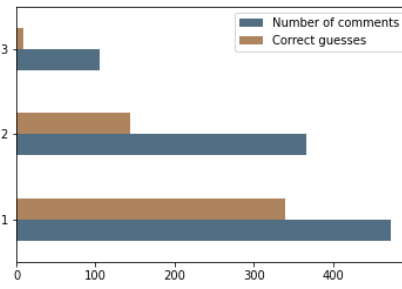


Fig 4.3: NBBOW3

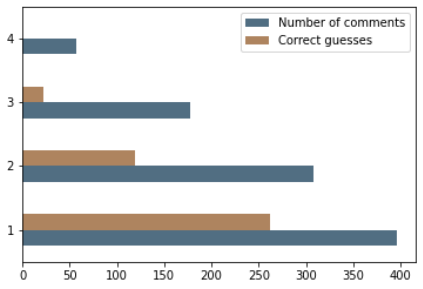


Fig 4.4: NBBOW4

Naive Bayes - Bag-of-bigrams

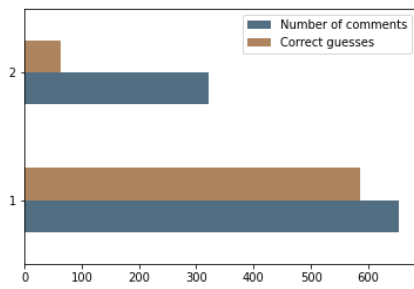


Fig 4.5: NBBOW2

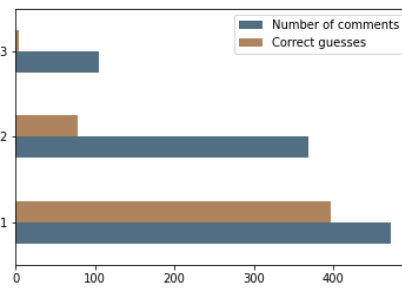


Fig 4.6: NBBOW3

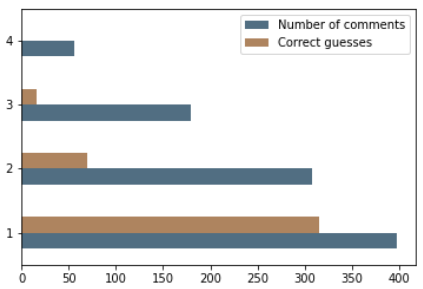


Fig 4.7: NBBOW4

Naive Bayes – TF-IDF

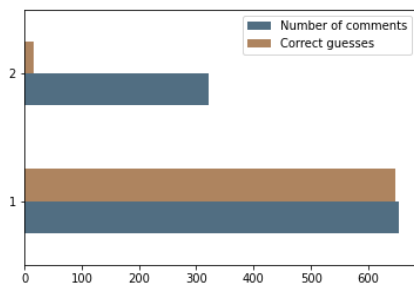


Fig 4.8: NBTfidf2

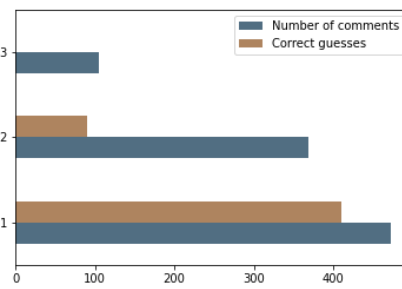


Fig 4.9: NBTfidf3

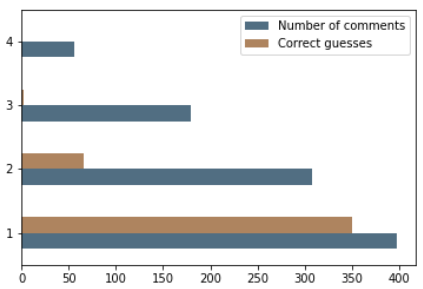


Fig 4.10: NBTfidf4

Naive Bayes – Engineered features

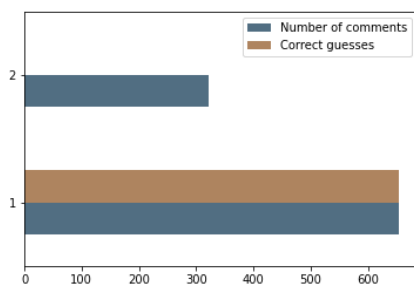


Fig 4.11: NBEF2

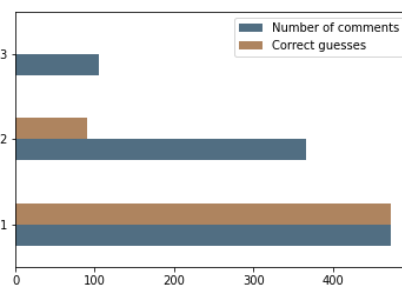


Fig 4.12: NBEF3

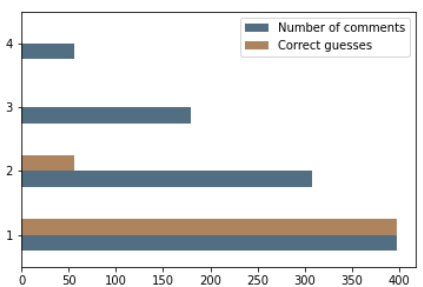


Fig 4.13: NBEF4

SVM – Bag-of-words

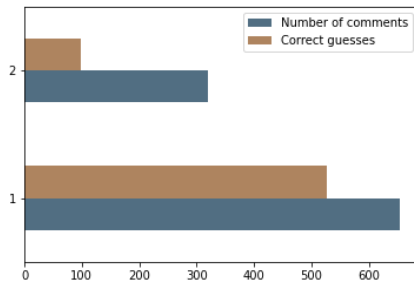


Fig 4.14: SVMBOW2

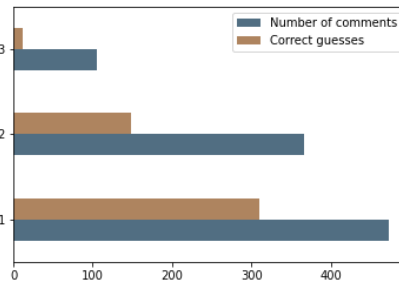


Fig 4.15: SVMBOW3

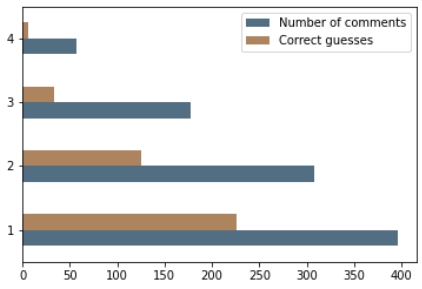


Fig 4.16: SVMBOW4

SVM – Bag-of-bigrams

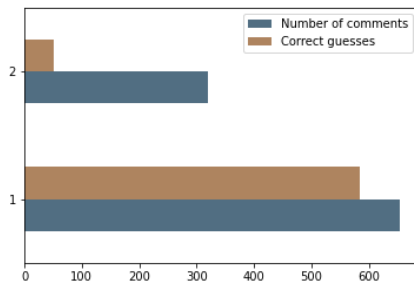


Fig 4.17: SVMBOB2

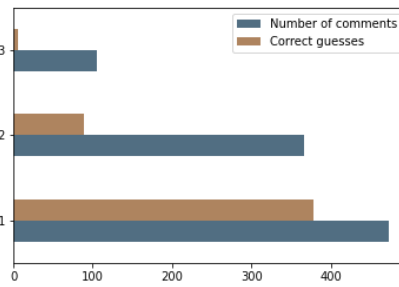


Fig 4.18: SVMBOB3

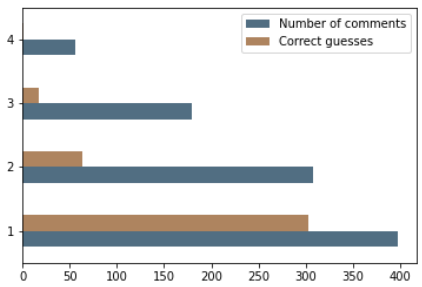


Fig 4.19: SVMBOB4

SVM – TF-IDF

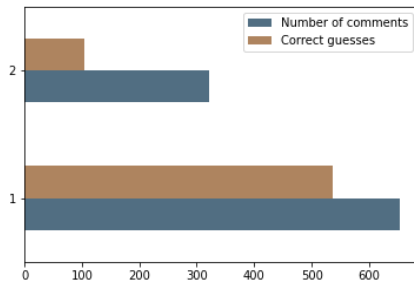


Fig 4.20: SVMTFIDF2

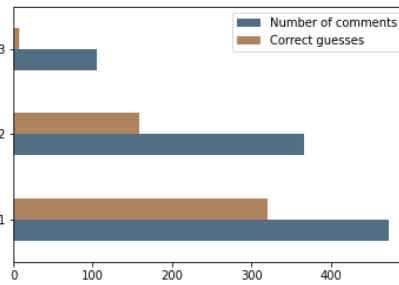


Fig 4.21: SVMTFIDF3

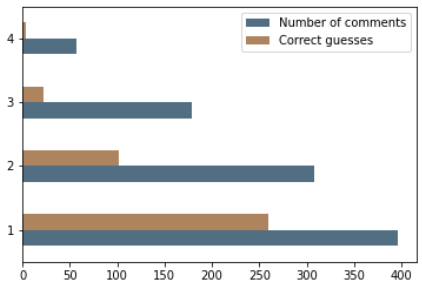


Fig 4.22: SVMTFIDF4

SVM – Engineered features

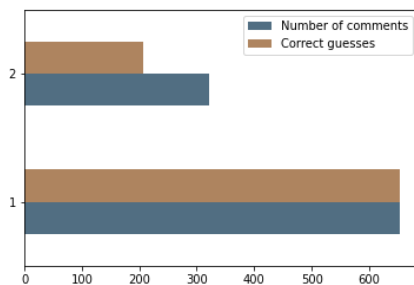


Fig 4.23: SVM EF2

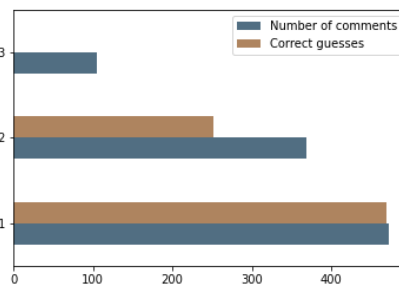


Fig 4.24: SVM EF3

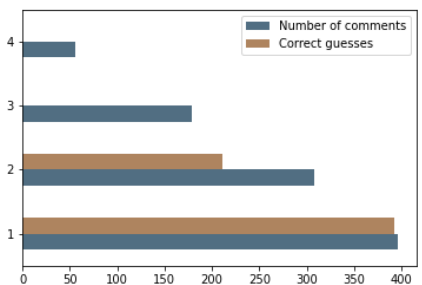


Fig 4.25: SVM EF4

4.1.1. Interpretation of the results

The names of each figure follow the pattern {MODEL}{FEATURE-SET}{BINS}, so, for example, the figure for the model in SVM with the EF and two bins is called *SVMEF2*.

The mean accuracy for NB is better than for SVM with all features except for the EF, which have much better accuracy than their NB counterparts. Note also that while the mean accuracy for SVM is lower, SVM has a higher accuracy for higher bars (meaning it was better at predicting high comments scores), as seen when comparing for example Fig 4.3. with 4.15 and 4.4 with 4.16 (and it's true for all 3-bin and 4-bin models in NB compared with all 3-bin and 4-bin models in SVM).

The EF have the highest mean accuracy for all models except for NB 2 bins, where TF-IDF did better by 0.25 per cent.

In 3- and 4-bin models the overall accuracy declines. However, the 4th bin and the 3rd bin in the 4-bin models sometimes hold higher score comments than the 3rd bin in the 3-bin models. For example, looking at the graphs for SVM – Bag-of-words, it is possible to see that the more bins the model used, the higher the accuracy for the higher comment scores. That means, if the purpose was to predict only whether a comment yields a high score or not, the 4th bin model in this case would have better accuracy.

4.2. Confusion matrices

Confusion matrices are presented for 14 of the models. Each matrix shows the number of predicted comment types compared to the number of actual comment types in a certain bin. The darker-coloured cells show the correct predictions and the lighter-coloured cells show missed predictions. Figure 4.26 shows a map of how to interpret the confusion matrices.

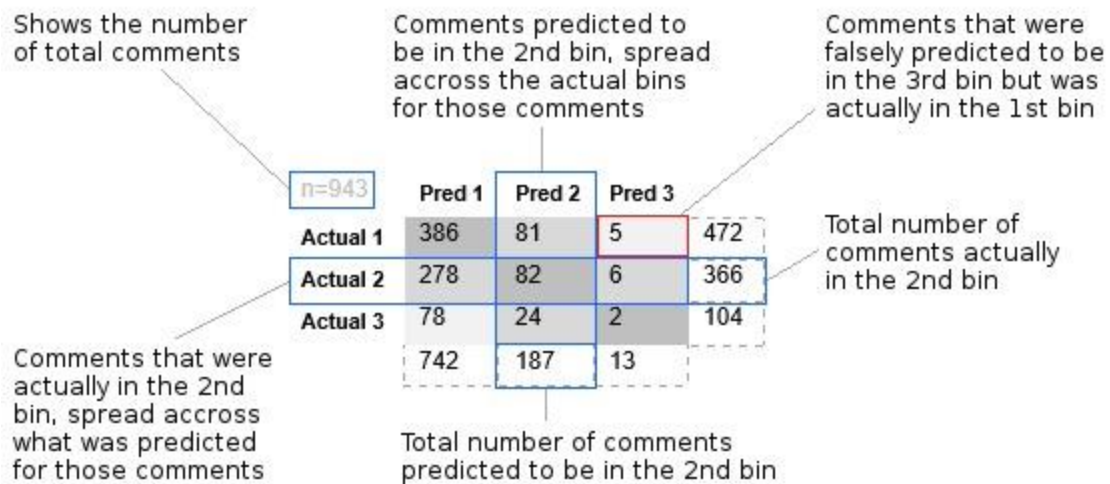


Fig 4.26: Confusion matrix interpretation

NB – Bag-of-words – 2 bins

n=968	Pred 1	Pred 2	
Actual 1	565	82	647
Actual 2	248	73	321
	813	155	

Table 4.2: NBBOW2

NB – TF-IDF – 2 bins

n=974	Pred 1	Pred 2	
Actual 1	645	9	654
Actual 2	310	10	320
	955	19	

Table 4.3: NBTfidf2

NB – Engineered – 2 bins

n=975	Pred 1	Pred 2	
Actual 1	653	0	653
Actual 2	321	1	322
	974	1	

Table 4.6: NBEF2

NB – Bigrams – 3 bins

n=943	Pred 1	Pred 2	Pred 3	
Actual 1	386	81	5	472
Actual 2	278	82	6	366
Actual 3	78	24	2	104
	742	187	13	

Table 4.4: NBBOB

NB – Engineered – 3 bins

n=943	Pred 1	Pred 2	Pred 3	
Actual 1	472	0	0	472
Actual 2	291	76	0	367
Actual 3	90	14	0	104
	853	90	0	

Table 4.7: NBEF3

NB – Bigrams – 4 bins

n=975	Pred 1	Pred 2	Pred 3	Pred 4	
Actual 1	314	65	16	1	396
Actual 2	222	64	20	1	307
Actual 3	129	34	13	1	177
Actual 4	42	11	2	0	55
	707	174	51	3	

Table 4.5: NBBOB4

NB – Engineered – 4 bins

n=975	Pred 1	Pred 2	Pred 3	Pred 4	
Actual 1	395	0	0	0	395
Actual 2	257	50	0	0	307
Actual 3	137	41	0	0	178
Actual 4	49	7	0	0	56
	838	98	0	0	

Table 4.8: NBEF4

SVM – Bag-of-words – 2 bins

n=975	Pred 1	Pred 2	
Actual 1	512	142	653
Actual 2	224	96	320
	735	238	

Table 4.9: SVMBOW2

SVM – TF-IDF – 2 bins

n=975	Pred 1	Pred 2	
Actual 1	531	123	654
Actual 2	230	91	321
	761	214	

Table 4.10: SVMTFIDF2

SVM – Engineered – 2 bins

n=975	Pred 1	Pred 2	
Actual 1	653	1	654
Actual 2	129	192	321
	782	193	

Table 4.13: SVMFEF2

SVM – Bag-of-bigrams – 3 bins

n=946	Pred 1	Pred 2	Pred 3	
Actual 1	376	84	13	473
Actual 2	275	79	14	368
Actual 3	78	23	4	105
	729	186	31	

Table 4.11: SVMBOB3

SVM – Engineered – 3 bins

n=946	Pred 1	Pred 2	Pred 3	
Actual 1	470	3	0	473
Actual 2	135	233	0	368
Actual 3	41	64	0	105
	646	300	0	

Table 4.14: SVMFEF3

SVM – Bag-of-words – 4 bins

n=975	Pred 1	Pred 2	Pred 3	Pred 4	
Actual 1	240	95	49	12	396
Actual 2	144	105	49	9	307
Actual 3	89	53	30	5	177
Actual 4	30	15	8	3	56
	30	15	8	3	

Table 4.12: SVMBOW4

SVM – Engineered – 4 bins

n=975	Pred 1	Pred 2	Pred 3	Pred 4	
Actual 1	391	4	0	0	395
Actual 2	119	187	0	0	306
Actual 3	61	116	0	0	177
Actual 4	23	32	0	0	55
	594	339	0	0	

Table 4.15: SVMFEF4

4.3. Sample comments

A sample of comments from the dataset with both correct predictions and wrong predictions is given below. The comments are presented together with their result for a specific model. The outcome is True if the model predicted the correct bin for the comment, or otherwise False.

#	Comment	Model	True/False	Predicted bin	Correct bin
1	All politics aside. She was an extraordinary human being who did extraordinary things. RIP Justice Ginsburg.	NBBOW2	T	1	1
2	I have multiple scars on my thumbs from slicing my thumb open on cans of ravioli and spaghetti-o's	NBBOW3	F	2	3
3	ENGLISH! I used to be terrible and my accent still sucks sometimes, but after more than 20 years of practice and a bachelor degree from an English university, now I'm quite proud to be bilingual.	NBBOW4	F	2	4
4	This is a little more niche, but Bill Wilson, the founder of alcoholics anonymous. The guy did great things, and created a program of recovery that has saved millions of lives since it's inception 85 years ago. He was also an arrogant ass hole that cheated on his wife even in sobriety. Edit: This isn't here to scare away anyone that's trying to get sober. I've been sober thanks to AA for 5 1/2 years. So don't let this keep you away if you're struggling with your drinking.	NBBOB2	T	2	2
5	Mom: Don't pinch that pimple now, you'll get a scar. Me: YOU CAN'T TELL ME WHAT TO DO, WOMEN! *gets scar*	NBBOB3	F	2	3
6	Not living my life to other peoples opinions/expectations. I go around barefoot, grow veggies and finally have chickens. Life is so good not being stressed out all the time. Let go of toxic people (and get a kettle, it'll change your life!)	NBBOB4	F	2	4
7	Mother Teresa.	NBTFIDF2	F	2	1
8	You laugh but having run into these fuckers in the wild I tell you, they would kill you and everyone you love.	NBTFIDF3	F	1	3
9	I'm not american, what does this mean for you guys?	NBTFIDF4	F	2	4
10	What next?? Haven't we had enough for 1 year already?	NBEF2	F	2	1
11	Spelling matters.	NBEF2	F	1	2
12	Pretty brutal really. My dad was attacked by an emu once. He is bald and his head is a little shiny. he was sitting on a bench at a wildlife park and The emu must have been attracted to the shiny reflection, snuck up behind him and bit his head. I have never seen someone jump so far in one leap before or since. As a child this traumatised me a little and now I am pretty uneasy around all kind of birds.	NBEF2	T	1	1
13	This is a little more niche, but Bill Wilson, the founder of alcoholics anonymous. The guy did great things, and created a program of recovery that has saved millions of lives since it's inception 85 years ago. He was also an arrogant ass hole that cheated on his wife even in sobriety. Edit: This isn't here to scare away anyone that's trying to get sober. I've been sober thanks to AA for 5 1/2 years. So don't let this keep you away if you're struggling with your drinking.	NBEF2	F	1	2
14	Understand how much more you will understand in the future and don't be ashamed to admit that you may not understand things.	NBEF3	F	1	3
15	Oldschool Runescape	NBEF4	T	3	1

Table 4.16: Sample sentences

#	Comment	Model	True/False	Predicted bin	Correct bin
16	What if Satan came down one day and said "It's pronounced 'Zlatan' then left?"	SVMBOW2	F	2	1
17	Covid cure. Then cute kittens.	SVMBOW3	F	1	2
18	Carol Baskin didn't kill her husband I believe he was heavily involved with drugs and thats the reason why everything he touched turned to 'gold'. I think he he flew drugs over to Mexico and it all went wrong and never made it back	SVMBOW4	F	1	4
19	Had a client at an animal hospital get upset when the veterinarian was a female. This lady legitimately threw a fit and refused to let the arguably much more qualified doctor see her pet for no other reason than that the vet was female. So, the doctor told her to get out. The lady wouldn't leave and started yelling at and berating the doctor, threatening to call the owner of the clinic. She even whipped out her phone and acted like she was going to dial. Now someone saying they know the owner is something that is very very common, as the owner is still a practicing vet well over 60 hours a week, and has been for 20+ years. Probably thousands of people have this guy's number. Most of them on the older side that try to flex it, though. So our female vet said go right ahead, I'm calling the cops if you don't leave. And that's exactly what happened. The client quickly left once the vet started speaking with the dispatcher. The police still came and took a report. Later when the owner caught wind, he left a note on her account that she was no longer welcome in our clinic :)	SVMBOB2	F	1	2
20	God is just a title to signify his level of power, his proper name is different depending on how you believe. Jehovah, Yahweh, Allah, Zarathustra, etc.	SVMBOB3	F	3	1
21	I woke up one night with my wife's hands in my mouth. When I sat up and asked what she was doing her reply was " I thought you were a vampire" 20 years later I still tease her about that one	SVMBOB4	F	3	4
22	"Oh jod"	SVMTFIDF2	F	2	1
23	"Oh jod"	SVMTFIDF3	T	3	3
24	parents and also video-games. I wish I could make a love letter explaining to every developer personally how they saved me through those dark times by creating media which started from simple arcade games to breathing worlds that also are cinematic masterpieces. Sometimes the gameplay makes up for it instead of the story while sometimes you need a bit of story with that gameplay. I'll just point out the main one. Valve, thank you for making media which helped me through dark times. I've played most of the games that you've made from l4d to cs 1.6...I can't thank you enough for bringing memories and a distraction to keep me away from that noose, even if you're a business I want you to succeed further since you've earned it. I loved the left 4 dead series and replayed portal & half-life multiple times over, not to mention the mods too. You deserved the break up until half-life alex and the fact that you even made hl:a gives me hope for the gaming industry. I love valve and I hope you do too!	SVMTFIDF4	F	3	4
25	I still to this day believe that for every sock that goes missing in your dryer it turns into a tupperware lid in you cabinet that fits nothing. I have seen no evidence to disprove this.	SVMEF2	T	2	2
26	FTL (faster than light), Darkest Dungeon, Terraria, Cardinal Quest 2 (mobile game), Rymdresa, Bomber crew Wow people need to play more indie games. Edit: I just looked at my stream library and have so many more to add. Unturned, dungeon of the endless, The binding of isaac, crypt of the necro dancer, realm of the mad God, spelunky, CASTLE CRASHERS!!	SVMEF3	F	2	1
27	I would be converted Edit: thanks for the upvotes and awards guys!=D	SVMEF4	F	1	2

Table 4.16: Sample sentences (cont.)

5. Discussion

Our 24 trained models are divided into three sections: the two models (NB and SVM), the binning delimiters, and the feature sets. Before we discuss these sections, we will briefly discuss the dataset and how its structure might have affected the results.

Dataset

An overview of the statistics for the dataset (3.1.1.) tells us that the number of significant words correlates with the number of stop words. The trend seems to be that more words equal lower scores. It might be because longer texts require more attention of the reader – it takes a longer time for the reader to conclude whether they like the comment or not. Comments #2, #3, #5, #9, #17 and #23 (in 4.3.), which were all labelled in the highest bin of their model, supports this idea. There are, however, some exceptions, as seen in comment #24. The dataset also shows that positive smilies can help the score, which we can see in comments #19 and #28. However, more observations could be made if we extracted more sample comments. It is the combination of features that is important, and that is too complex for us humans to see, which is why we use machine learning models.

Models

An interpretation of the models' results has been given in section 4.1.1. With as many as the 6000 most *interesting* features (defined by highest TF-IDF) for the SF, NB does a little bit better in comparison to SVM. Our EF has only 11 features, and the fact that these features have a smaller size probably impacts the results in the SVM models. We cannot describe exactly why one model performs better than the other without delving into mathematics, which is not in focus for this thesis. However, a guess as to why SVM performs better than NB for our EF is that SVM performs better at larger scales, while NB usually performs better with text snippets (Wang & Manning, 2012).

Binning

Binning data can also have an impact on the results. We can see that with the bigger number of bins, the average accuracy for the entire model decreases, but the accuracy for some models in the higher spectrum improves. When moving from 2-bin models to 3-bin or 4-bin models, we allow the models to shape the comments to their real-world labels better. “Popular” and “unpopular” are binary values which we humans tend to think of, but these labels rarely model the real world (there would be more popularity labels, indicating various levels of popularity). When moving to higher-bin model, we could allow for a third label, perhaps “not considered”

(as though the readers scrolled past the) and a fourth “not seen”. This is apparent when looking at comment #22 and #23. The SVMTFIDF2 predicts that this same comment belongs to the first bin and SVMTFIDF3 predicts that it belongs to the third bin. This might mean that SVMTFIDF3 recognises the lower level comments in a more fine-grained way and therefore rightly predicts this comment to be a popular one. We can also see this in confusion matrices for SVMEF2 and SVMEF3 in Table 4.13 and Table 4.14, respectively. NBBOB3 and NBBOB4 in Table 4.4 and Table 4.5. also shows this. Perhaps, 13 correct predictions in NBBOB4 and 2 correct predictions in NBBOB3 corresponds with our real-world label “popular” – in this sense, then, NBBOB4 outperformed its lower binned counterpart NBBOB3.

Features

There are three reasons for selecting the engineered features that we have:

- They are easy to craft and replicate
- They work well with one another in different sets of combinations
- They are powerful for popularity classifications

The first point is related to our research goal. The second point will be discussed below, and the third point will be clarified by the end of this chapter and covered in the Conclusion.

We will explain this by (1), discussing that SF does not necessarily take context, semantics, or pragmatics into account. Then, by (2), we discuss why these points are crucial for predicting popularity, and a short explanation for why SF still do reasonably well will also be given. Ultimately, by (3), we will make the case that our EF do care about context and semantics and therefore perform better than SF for the task of popularity prediction using NB or SVM.

- (1) Bag-of-words feature representations have been shown to be a good set of representations for sentiment analysis (El-Din, 2016). The reason might be that when a sentence gets classified, the bag-of-words-style representations put a weight for the words in that sentence towards the class it represents. When these words are encountered in a new sentence again, no matter the context, the new sentence will be weighted towards that same class by those words (Table 2.2 shows this reasoning well).

Let us look at the sample comment #8 in table 4.16, which contains a bad word. From a sentiment point of view, it is possible to use prior evidence to see whether that word seems to be popular. It works because when breaking down a sentence to its words, it is possible to predict which level of ‘popularity’ those words usually entail from prior data. However, the

judgement is passed down in a contextless sense. It does not matter to the algorithm if the word was a bad word or not.

Bag-of-words does not discern for semantics or pragmatics. This can be proven in an example of thought by first substituting the words with other symbols, i.e., numbers. Secondly, by replacing the features in the feature table and doing the calculations for Naive Bayes algorithm (or SVM), the result for new sentences (again substituting the words) would be the same. Therefore, the words themselves act only as a placeholder for an encounter of meaningless symbols, whatever those symbols might be.

- (2) When predicting popularity, however, meaning and context are crucial. This is because a written text is popular or unpopular based on how it is perceived (and in the Reddit case voted on) by its readers. Others can agree, disagree, or make associations with the text regardless of the sentiment from the author.

When studying our Reddit comments and their popularity, it becomes clear that prior knowledge of the words does not say much about the comments level of popularity - even though the word can be marked having a low popularity in one context, it does not necessarily mean that it is negative in another. The word can still have a good chance of getting positive feedback in another context. A general example of this can be seen in a political debate - a given sentiment in a particular topic might be negatively received from one party, whereas it can be regarded as positive from the opposing party.

Perhaps bag-of-words does not work at all for checking popularity - it might only do so well because we are confined to a small sample of reality in our tests. With our political parties as an example, it might be that we only received instances of discussions from one side of the party in our 16840 comments - but if we were to expand the dataset, maybe we would see a decline for all bag-of-words statistics.

The same is true when looking at TF-IDF (see 2.1.6.); it does not matter how we count the words (be it by the occurrence or mathematical formula) when the words are only symbols without meaning or concern for context. Bigrams take context into account a little, but that is a “local context” (i.e., the sentence itself) where we instead of checking the words check for longer parts of a comment (see 2.1.4.). However, for a more “social context”, it misses the meaning for what one side values and the other does not (in the example of our political debates and the parties).

- (3) This is where our EF excel. We will proceed by presenting several accounts where our EF take context, semantics, or pragmatics into account.

First, we cover the similarity of the submission as well as the similarity with other comments. (See details in section 3.3.1.). In our political debate example, a sentence here will only do good if it resonates with its context, that is if the sentence is received well from the first party or the opposing party. These features do not give the words any meaning per se - a bad word can be popular in one context and unpopular in another, if it resonates with its context, i.e. when viewed from the different parties. These features take the topic and the other authors comment into account. That means if the topic itself is popular, or most of the comments are popular, only a similar comment will be regarded as good and receive popularity of its own. A comment that strays away from the topic or the other authors' comments (i.e., in this political debate) will probably be considered ill and be rendered unpopular.

Secondly, EF analyses words by their meaning at a lexical level (stop words vs significant words) and at a semantic level (bad words). These words use a general context here and they are not meaningless. For example, a bad word is always a bad word in our EF.

Thirdly, EF counts the length of the comment (by derivation of the count of the different words, e.g. *word types*) and whether it contains a link or a name. This is meaningful because the length of the text and the time it takes to read it correlates directly with how well it is perceived. It also looks at the ratio between stop words and significant words. Lower stop word ratio helps convey meaning in text (Rahman et al., 2018). The link and names combined can be used to see if a comment references a work or in any other way backs up the author's claims, i.e., in a political (or for example scientific) debate.

Lastly, it has been shown that smilies do help convey the sentiment of texts (Kralj Novak et al., 2015). While a sentence could include bad words or fail to resonate with the "social context", a smiley might be used to indicate sarcasm or perhaps sadness about a stated fact, which can reverse the meaning of the comment. We also believe that sentiment might affect the level of popularity that a particular comment has, therefore, we use such features in our research.

Engineered features might outperform SF for specific tasks when analysing and understanding the text's medium and how it behaves on it. However, SF are still a good choice because they are generic, common and known in the field. They also have the advantage to be quickly developed for an experienced person because there are many standard tools that support them.

5.1 Future research

The selected features in our EF set have some limitations. One of them is that all EF are morphological. We do argue (as per earlier statement) that context and meaning are crucial for popularity prediction and that the engineered similarity features take context in consideration, however, all our EF work by counting tokens. It is only when using the features together that properties such as semantics and pragmatics emerge. For future research, it would be useful to advance in the linguistic pyramid from the morphological level towards the pragmatic meaning of text by adding features which by themselves measure these properties.

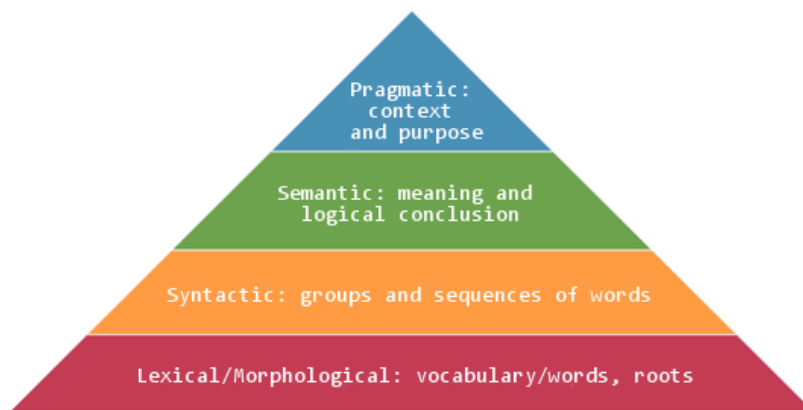


Fig 5.1: The linguistic pyramid

It is possible to engineer new feature extraction algorithms with the help from standard tools in NLTK (Bird et al., 2009), or engineer brand new feature extraction algorithms. NLTK provides a means to generate phrase structure trees and part of speech-tagging. New features could perhaps be used to decide the active or passive form of a sentence, or determine the tense (or maybe even Reisenbach tense), which might infer a lot of information. For example, in various discussion categories, one's own experience and reflections are not welcomed or just not a popular form of writing (as in a scientific forum where stated facts would typically receive a higher score). Own reflections or anecdotes are usually given in past tense (or the posterior past in Reisenbach tense). Numerous syntactic features could help in this field. Semantic features might also be helpful. Moschitti and Basili (2004) pursued to use senses in WordNet to group words by semantic meaning, although this had no significant effect. Future research could help show which features would help in the task of popularity prediction as well as in other text classification tasks.

6. Conclusion

In this thesis we examined how specific engineered linguistic features can be used for better accuracy and results in popularity prediction of texts with NB and SVM classifiers. The results indicate that, in general, the algorithms benefit from our manually constructed features. In particular, SVM benefits from the use of EF the most. We propose that our EF take textual context into account (at least to some degree). Additionally, the EF provide the advantage of being easy to understand. Lastly, this research can be helpful for linguists who want to start working with automatic classification tools for their data or perhaps want to create their own engineered features.

References

Aramaki, E., Maskawa, S., Morita, M., **2011**. Twitter Catches The Flu: Detecting Influenza Epidemics using Twitter. Edinburgh, Scotland, UK. *Association for Computational Linguistics*. Pages 1568–1576. Available at: <https://www.aclweb.org/anthology/D11-1145/> (accessed 16th December 2020)

Bad Words List and Page Moderation Words List for Facebook, **No date**. *Free web headers*. Available at: <https://www.freewebheaders.com/bad-words-list-and-page-moderation-words-list-for-facebook/> (accessed 12th December 2020)

Baer, D., **2020**. The 31 biggest subreddits. Social Media Marketing Resources, *OneUp Blog*. Available at: <https://blog.oneupapp.io/biggest-subreddits/> (accessed 16th December 2020)

Bayes, T., **1763**. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S. *Philosophical Transactions of the Royal Society of London*. Vol 53. Pages 370–418. DOI: <https://doi.org/10.1098/rstl.1763.0053>

Boe, B., Rees-Hill, J., Roth, L., Payne, J., Goodman, R., Goankar, V., **2016**. Python Reddit API Wrapper Development. PRAW. Available at: <https://github.com/praw-dev/praw> (accessed 16th December 2020)

Bird, S., Klein, E. & Loper, E., **2009**. Natural language processing with Python: analyzing text with the natural language toolkit, *O'Reilly Media, Inc*. Available at: <https://www.nltk.org/> (accessed 16th December 2020)

Boser, B., Guyon, I., Vapnik V., **1992**. Support Vector Machines. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*. 5. Vol 5. DOI: <https://doi.org/10.1145/130385.130401>

Cherbas, C., **1992**. Natural language processing, pragmatics, and verbal behavior. *The Analysis of verbal behavior*. Vol 10. Pages 135–147. DOI: <https://doi.org/10.1007/BF03392880>

Demonstrating the different strategies of KBinsDiscretizer, **2020**. Available at: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_discretization_strategies.html (accessed 12th December 2020)

El-Din, D., **2016**. Enhancement Bag-of-Words Model for Solving the Challenges of Sentiment Analysis. *International Journal of Advanced Computer Science and Applications*. Vol 7. DOI: <https://doi.org/10.14569/IJACSA.2016.070134>

Foufi V., Timakum T., Gaudet-Blavignac C., Lovis C., Song M., **2019**. Mining of Textual Health Information from Reddit: Analysis of Chronic Diseases With Extracted Entities and Their Relations. *J Med Internet Res*. Vol 21. e12876. DOI: <https://doi.org/10.2196/12876>

- Jurafsky, D., Martin J.H., **2020**. *Speech and Language Processing, 3rd edition [online]. N-gram Language Models, Naive Bayes and Sentiment Classification. Chapter 3 & 4*. Available at: <https://web.stanford.edu/~jurafsky/slp3/> (accessed 16th December 2020)
- Kralj Novak P, Smailović J., Sluban B., Mozetič I., **2015**. Sentiment of Emojis. *PLOS ONE*. Volume 10, e0144296. DOI: <https://doi.org/10.1371/journal.pone.0144296>
- Leppänen, D., **2020**. Parseit. Available at: <https://github.com/halpdesk/parseit> (accessed 16th December 2020)
- Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P.S., He, L., **2020**. A Text Classification Survey: From Shallow to Deep Learning. *IEEE Transactions on Neural Networks and Learning Systems*. Volume 3, no 11. Available at: <http://export.arxiv.org/pdf/2008.00364> (accessed 12th January 2021)
- Lin Y., **2020**. 10 Reddit statistics every marketer should know in 2020. *Oberlo*. Available at: <https://www.oberlo.com/blog/reddit-statistics> (accessed 12th December 2020)
- Manning C.D., Raghavan P., Schütze, M., **2008**. Introduction to Information Retrieval. *Cambridge University Press*. Page 260. Available at: <http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf> (accessed 16th December 2020)
- Moschitti, A., Basili, R., **2004**. Complex Linguistic Features for Text Classification: A Comprehensive Study. Berlin, Heidelberg. *Springer Berlin Heidelberg*. Pages 181-196. DOI: https://doi.org/10.1007/978-3-540-24752-4_14
- Mosteller, F., Wallace, D.L., **1963** Inference in an Authorship Problem. *Harvard University, Center for Advanced Study in the Behavioral Sciences, USA*. Vol 58, No 302. Pages 275-309. DOI: <https://doi.org/10.1080/01621459.1963.10500849>
- Nothman, J., & Qin H., & Yurchak, R., **2018**. Stop Word Lists in Free Open-source Software Packages. *Association for Computational Linguistics. Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. Pages 7-12. DOI: <http://dx.doi.org/10.18653/v1/W18-2502>
- Pedregosa, F. et al., **2011**. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct), pp.2825–2830. Available at: <https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html> (accessed 12th December 2020)
- Pranckevičius, T., Marcinkevičius, V., **2017**. Comparison of Naive Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification. *Baltic Journal of Modern Computing*. Vol. 5. Pages 221-232. DOI: <http://dx.doi.org/10.22364/bjmc.2017.5.2.05>
- Rahman, M. M., Roy, C. K., Kula R. G., **2017**. Predicting Usefulness of Code Review Comments Using Textual Features and Developer Experience, *IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, Buenos Aires. Pages 215-226. DOI: <https://doi.org/10.1109/MSR.2017.17>

reddit.com Competitive Analysis, Marketing Mix and Traffic, **No date**. *Alexa*. Available at:
<https://www.alexa.com/siteinfo/reddit.com> (accessed 12th December 2020)

Spräck Jones , K., **2004**. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*. 28, 11-21, 1972 and 60, 493-502,. DOI: <https://doi.org/10.1108/eb026526>

Subramanian, S., Baldwin, T., Cohn, T., **2018**. Content-based Popularity Prediction of Online Petitions Using a Deep Regression Model. Association for Computational Linguistics. *Association for Computational Linguistics*. Vol 2. Pages 182–188. DOI: <http://dx.doi.org/10.18653/v1/P18-2030>

Wang, S., Manning, C.D., **2012**. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *ACL ' 12*. Volume 2. Pages 90-94. Available at:
https://nlp.stanford.edu/pubs/sidaw12_simple_sentiment.pdf (accessed 16th December 2020)